A PRIMERA REVISTA DE PROGRAMACION EN CASTELLANO

SOLO PROGRAMADORES

O III, NÚMERO 33

975 Pts.

Formato audio MPEG a fondo

WWW: JavaScript

El modelo de objetos

PROGRAMACIÓN DELPHI

Propiedades y eventos

REDES LOCALES

Gestión de rendimiento en Novell

TECNOLOGÍA MULTIMEDIA

MMX: Instrucciones y aplicaciones

- Java
- C++ Builder
- Visual Basic avanzado

CONTENIDO DEL CD-ROM

- JDK 1.1.1 y tutoriales
- Componentes Delphi
- Utilidades MPEG
- Demos de Visual J++ y C++ Builder

00033
8 413042 303299

SÓLO PROGRAMADORES

Número 33
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es

Directora Adjunta
Amelia Noguera
anoguera@towercom.es

Editora-Redactora
Francisca Guzmán

Colaboradores
Enrique Díaz, Miguel González Maestre, Arsenio Molinero, Carlos Álvaro, Enrique de la Lastra, Francisco José Abad Cerdá, Jose Antonio Collar, Javier Sarsa, Oscar Prieto Blanco, Ernesto Schmitz, Alejandro Reyero, Emilio Postigo Rian, Chema Álvarez.

Jefe de Diseño
Fernando García Santamaría

Maquetación
Clara Francés

Tratamiento de Imagen
María Arce Giménez

Imagen de Portada
Fernando Núñez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11

Publicidad
Papín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Isabel Bojo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Filmación
Filma Dos S.L.

Impresión
G. Reunidas

Distribución
SGEL

Director General
Antonio M. Ferrer Abelló

Director Financiero
Francisco García Díaz de Liaño

Director de Producción
Carlos Peropadre

Directora Comercial
Carmina Ferrer

Director de Distribución
Enrique Cabezas García

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792



Palabra de informático

Lo prometido dicen que es deuda. Y nuestra deuda está saldada. Podéis contemplar una nueva imagen de la revista, renovada para estar a la altura de los contenidos, y también vamos a haceros caso y en este número no vais a encontrar los cartones que tanto os molestaban. Además volveréis a ver algunas de las secciones que nos sugeristeis que continuáramos.

Nos pedís nuevos temas, nuevos enfoques y nosotros los buscamos. Sin embargo, para aquellos más conservadores que echan de menos lo que ya incluíamos, tenemos unas palabras: nada ha desaparecido, la revista está abierta a nuevas perspectivas, pero conserva los contenidos que demandáis con vuestras cartas, con vuestros mails. Visual C++, IA, el curso de Visual Basic, continuarán en próximas entregas. No es fácil; cada mes nos enfrentamos a la necesidad de valorar qué artículos salen, teniendo en cuenta el momento y vuestras peticiones. Pero mantenemos nuestra promesa. En este número se reanuda el curso de JavaScript y continuamos con la sección de Delphi, que muchos habéis alabado.

Y a esto llegamos, os debemos un sincero agradecimiento a todos los que os interesa tanto la informática que os tomáis la molestia de escribir unas palabras para expresar vuestra opinión. Y por eso queremos daros las gracias.

No dejéis de hacerlo, porque vuestra ayuda nos anima a continuar mejorando.

Siempre podéis utilizar la dirección de correo electrónico de Sólo Programadores: solop@towercom.es para dirigir vuestras sugerencias. Pero también para criticar lo que no os guste y para enviar vuestras dudas que se contestan primero vía e-mail y después desde las páginas de la revista, para orientar a otros que pudieran tener las mismas.

Sugerid productos que queráis que incluyamos en el CD, shareware, tutoriales, todo lo que necesitéis en vuestros estudios o en vuestro tiempo libre, y temas, porque seguimos buscando expertos para que nos enseñen lo que ellos saben.

A todos se os contestará debidamente. Palabra de directora.

Y espero que os guste el artículo sobre el formato de compresión de audio MPEG Layer3 y que utilicéis todas las herramientas que os proporcionamos sobre el mismo. Porque creemos que puede ser importante. Que disfrutéis descubriéndolo.


33



CONSULTAS, SUGERENCIAS, CRÍTICAS, COMENTARIOS

Tenéis a vuestra disposición la dirección de correo electrónico:

solop@towercom.es

Todos los mensajes que nos lleguen serán contestados.



Noticias

PRODUCTOS PARA INTERNET

De nuevo se establecen alianzas para obtener mayor competitividad en Internet

Informix Software, Inc. Y Hewlett-Packard Company integrarán sus tecnologías para responder a la continua evolución de Internet. Quieren conseguirlo integrando la gama HP Domain XE de la familia HP Domain Enterprise Server (Servidores Empresariales de dominio) con Informix-Universal Web Architecture. Este tecnología de Informix permite la gestión de contenidos dinámicos además de los tradicionales.

La incorporación de Universal Server de Informix permite combinar el rendimiento y la escalabilidad de la arquitectura DSA con la tecnología DataBlade (para más información sobre esta tecnología consultar Sólo Programadores nº 32). A partir de los módulos DataBlade los servidores pueden manejar datos complejos entre los que se incluyen páginas Web, números, imágenes, vídeos, series cronológicas y sonido, entre otros.

El servidor incluye Universal Web Connect de Informix, permitiendo así a los usuarios escribir aplicaciones para Internet. Con los módulos Web DataBlade y TextDataBlade de Informix se consigue gestionar los contenidos y las aplicaciones fácilmente, así como realizar búsquedas por el contenido.

Para más información: Natividad de Mateo (91) 3 72 98 00 de Informix Software Ibérica.

INFO.COM/97

Se celebró los días 8, 9 y 10 de abril, organizada por Microsoft y Telefónica.

Eran todos los que estaban y estaban muchos de los que eran. Más de 30 empresas del sector informático entre los que se incluyeron distribuidores, desarrolladores, fabricantes de PCs y proveedores de acceso a Internet montaron sus stands y redoblaron sus esfuerzos durante tres días para convencer a las empresas de la necesidad de informatizarse.

Con este objetivo en mente, empresas como Microsoft, Telefónica, Digital, Sun, BT Systems, entre otras muchas, mostraron a todo el que quiso acercarse al Palacio de Congresos y Exposiciones de Madrid (la entrada era gratuita) sus últimas tecnologías de comunicación.

Así se pudieron ver soluciones para todos los gustos para el acceso a Internet e Infovía, así como las nuevas plataformas de 32 bits.

Para completar la información que en los stands se proporcionaba sobre los productos de cada una de las empresas, se dieron una serie de conferencias que permitieron a los asistentes entender un poco más las peculiaridades de la amplia oferta.

MICROSOFT WINDOWS NT SERVER 4.0 INCORPORA LA TECNOLOGÍA STEELHEAD

De esta forma se añaden capacidades de enrutamiento e interconexión de redes

Steelhead es un conjunto de tecnologías de interconexión de redes y enrutamiento, diseñadas para añadir servicios avanzados de comunicación a Microsoft Windows NT Server; se integra en el sistema operativo y se instala como un servicio de acceso remoto y servicio de enrutamiento combinado, de fácil utilización. Además es abierto y extensible con API's que permiten a los desarrolladores crear soluciones de enrutamiento a la medida.

Las características de esta tecnología son:

- Interfaz de usuario gráfico e interfaz de línea de comandos para el scripting.
- Apis para los protocolos de enrutamiento, interfaces de usuario y gestión de otras compañías.
- Conjunto completo de protocolos de enrutamiento para IP e IPX (incluido OSPF de Bay Networks).
- Servidor a Servidor PPTP para Redes Privadas Virtuales seguras.
- Enrutamiento de "marcación bajo demanda" (demand-dial).

Para más información sobre la tecnología o para acceder a su segunda beta, puede conectarse a:

http://www.microsoft.com/ntserver/info/steelhead.htm

NUEVAS SOLUCIONES INTRANET

Netscape y Novell se unen en un Acuerdo Tecnológico

A través de la compañía de nueva creación y capital conjunto de las dos empresas, Novonyx, situada en Utah, Netscape y Novell colaborarán en la creación de Intranet y Extranet. La idea es aunar sus tecnologías para comercializar la familia de software de productos de Netscape SuiteSpot sobre la plataforma de red Novell IntranetWare.

IntranetWare es la plataforma Novell orientada a Intranets actuales, que se soporta sobre NetWare 4, y que ofrece soporte completo de TCT/IP, directorio, seguridad de red, conectividad, gestión global, publicación Web y servicios de ficheros e impresión a redes dentro de la empresas; Netscape SuiteSpot es una suite de servidor que incluye herramientas de publicación, correo electrónico y groupware en las Intranets, con productos como Netscape Messaging Server, Netscape Enterprise Server, Netscape Collabra Server y el entorno de desarrollo LiveWire Pro.

Breves

INCREMENTO DE RENDIMIENTO EN LA CONMUTACIÓN LAN

3Com anuncia la oferta de Gigabit Ethernet para soportar mayores anchos de banda

Las nuevas soluciones de 3Com proporcionarán velocidades Gigabit en cualquier punto de la red, permitiéndole operar con un alto rendimiento 10/100/1000 Mbps mediante los conmutadores SuperStack II y CoreBuilder y las tarjetas EtherLink. En la nueva oferta se incluyen High-Function and Boundary Switches y una nueva tarjeta de interfaz de red para backbone, cableado y conectividad en el servidor.

Para un futuro, 3Com anuncia su intención de ofrecer soluciones de enrutamiento Gigabit Ethernet y Fast Ethernet LAN, que soporten casi 30 millones de paquetes por segundo para la primera mitad de 1998.

Para más información: FUNCORP CONSULTING, Cristina Buraya (91) 542 80 90

TEAMWARE OFFICE 5.1

Suite de groupware para intranet de Teamware

El entorno de groupware para intranet desarrollado por Teamware ofrece una forma para tener correo electrónico Internet y colaboración, y se adapta a las pequeñas y medianas empresas como una solucín asequible y a las grandes empresas por sus características de escalabilidad y estabilidad.

Como características principales tiene la conectividad X.400(88), la integración de directorios LDAP/X.500 y sus características de búsqueda de textos avanzadas.

NUEVA GENERACIÓN DE KNOWLEDGE DISCOVERY

Arquitectura Data Minig Relacional orientada a objetos

En un intento por integrar el software data minig con una base de datos relacional, Tandem anuncia la tecnología Object Relational Data Minig. Esta tecnología hace posible una mejora en el conocimiento de los datos que se poseen para proporcionar soluciones en márketing de tarjetas de crédito, peticiones de análisis y análisis de detalle en el sector seguros para mejorar costes y los ratios de pérdidas, detectando fraudes y, en otros procesos encuadrados en el comercio minorista como son la extracción de volúmenes de datos muy grandes.

Los datos del almacén están disponibles en tiempo real, lo cual significa un tiempo de procesamiento menor y una gestión más sencilla. A partir de un interfaz SQL entre las herramientas data minig del usuario y los mecanismos objeto relacionales y de base de datos relacional, se realizan las funciones de manipulación de datos especializadas que requieren los algoritmos de data minig.

Esta tecnología permite comprender de una forma rápida un mayor número de modelos de datos, como voz, vídeo, imagen y otros objetos multimedia. Para más información utilicen la dirección de correo: firstlinepress@mad.servicom.es

AUMENTA LA DEMANDA DE PROFESIONALES EN ÚLTIMAS TECNOLOGÍAS

Según un estudio realizado por la Empresa de Servicios de Formación Informática BIT, el sector de las "Tecnologías de la información" incrementó la demanda de profesionales un 75% durante 1996. El estudio fue realizado tomando como base los anuncios aparecidos en la sección de demandas de empleo del periódico "La Vanguardia".

Para los informáticos es la mayor tasa de demanda, con un aumento del 125% respecto a 1995, tanto profesionales informáticos como profesionales de otros sectores a los que se les requiere conocimientos de informática.

En cuanto a los puestos de trabajo requeridos, se destaca un crecimiento generalizado en todos, si bien los cargos como directores, jefes y responsables, programadores y analistas, técnicos de sistemas y comerciales son los que se llevan la palma. También aumentan las demandas de aquellos puestos que requieren conocimientos informáticos.

Los lenguajes también han sido objeto de estudio, los más solicitados son los lenguajes visuales, Visual Basic, PowerBuilder, Delphi y Java, y se afirma en este estudio que éstos son los lenguajes del futuro. En cuanto a los sistemas operativos, se destaca Windows, Unix y MS-DOS.

Para más información consultar la dirección http://www.bit.es

NUEVO API PARA APLICACIONES CAD/CAM/CAE

El API, llamado OpenGL Optimizer, es una interfaz de programación de aplicaciones de nueva generación ideado para mejorar las prestaciones de visualización en CAD/CAM/CAE. El Open GL es un API de gráficos 3D flexible y estable que permite a los desarrolladores de software para supercomputación, estaciones de trabajo y PCs, la creación de imágenes en color y objetos 3D de calidad.

Se creó para el desarrollo de aplicaciones de rendering, de forma independiente de la plataforma y proporciona funciones gráficas 2D y 3D como son transformación, color, modelado, iluminación y sombreado suave. Además permite la obtención de características como mapeado de texturas, niebla, alpha blending y motion blur.

El producto permite que las aplicaciones hagan uso de todas las prestaciones del hardware de gráficos de su máquina sin preocuparse del tipo de plataforma, ya que los elementos de la librería son códigos estándar c++ que operarán sobre distintas plataformas, incluyendo PCs con WindowsNT, Windows 95 y las estaciones de trabajo UNIX.

El API proporciona características de rendering de ray-tracing de forma interactiva, además de simplificación, teselación avanzada y generación automática de niveles de detalle. En conjunción con una carga y una paginación de la base de datos más rápidas el API es un intento por conseguir una base sobre la que definir un API abierto y estándar de la industria para aplicaciones CAD/CAM/CAE.

Para más información dirigirse a la dirección de correo electrónico: firstlinepress@mad.servicom.es

TECNOLOGÍA PROJECT STUDIO DE SUN

Novell pone la tecnología Java al alcance de los usuarios de IntranetWare

La compañía licenciará la tecnología de desarrollo en Java Project Studio de Sun Microsystems, Inc. Y se integrará con versiones de IntranetWare, la plataforma de Novell de intranet y acceso a Internet.

Project Studio permitirá que los usuarios puedan crear soluciones Java de Internet e intranets sin tener que escribir programas o scripts. El entorno de desarrollo visual de Project Studio utiliza la arquitectura de componentes JavaBeans, lo cual permite a personas sin conocimientos de programación crear programas a partir de bloques de código preescrito llamados "beans". La tecnología Visual Java de Project Studio y la arquitectura de componentes de JavaBeans permitirán que diferentes usuarios puedan crear rápida y fácilmente aplicaciones de Intranet e Internet sin tener que escribir una sola línea de código.

Para más información consultar: http://www.sun.com o http://www. novell.com

CISCO SYSTEMS, INTEL Y MICROSOFT CREAN NETWORKED MULTIMEDIA CONNECTION

Su objetivo es implantar y promover los estándares de la industria y acercar las aplicaciones multimedia en red a empresas que trabajan con intranets o Internet.

El programa NMC supone la colaboración de las tres compañías para facilitar a los desarrolladores la creación de aplicaciones como la enseñanza interactiva, la edición de información y la videoconferencia y para ayudar a las empresas a que puedan implantar este tipo de aplicaciones.

A través del programa las tres compañías ofrecerán soporte y recursos para los desarrolladores de aplicaciones multimedia, los suministradores de servicio y los directores de Tecnologías de la Información. Con dicho fin están trabajando junto con fabricantes de software independiente, suministradores de servicio de Internet, integradores de sistemas y compañías de tecnologías de la información que suministran a los usuarios finales soluciones informáticas. Para ello ofrecerán:

- Un laboratorio de tecnología multimedia, instalado en las instalaciones de Cisco para el desarrollo y análisis de la interoperabilidad.

- Nuevas herramientas de gestión y servicios de implantación dirigidos a facilitar la adopción por parte de los servicios de tecnologías de la información; estas herramientas incluyen SDKs de las tres compañías, código, utilidades hiperenlaces y documentación.

Breves

OFERTA PARA LAS EMPRESAS QUE QUIERAN ADENTRARSE EN INTERNET

IBM, Oracle y Netscape ofrecen en conjunto una solución para Internet

IBM pone su IBM/ RS/600, un modelo 43P, con procesador PowerPC 604e a 166 MHz, 32 MB de RAM, 2,1 Gb de disco y sistema operarivo AIX 4.1, adaptador gráfico PCI Power GXT110 y conexión a Internet. Netscape pone el software SuiteSpot, que incluye los servidores Proxy, News, Catalog y Mail, además de Live Wire, y Oracle se encarga del Workgroup Server y Webserver para AIX.

Las empresas distribuidoras que se acojan a esta campaña serán Solution Provider de IBM RS/6000 y Partner Autorizado de Netscape y Oracle. Para más información en IBM: 900 100 400.

La oferta incluye un curso de formación sobre los productos de la misma y su correspondiente soporte técnico.

WINFAX PRO 8.0

Nueva versión del software de fax

Esta aplicación funciona con Windows 95 y Windows NT (3.51 y 4.0) y proporciona una forma sencilla de realizar las tareas de fax más comunes sin tener que cargar todo el programa. Además dispone de un asistente para la creación de páginas de cubierta y más macros para Word y Excel para Office ´97.

WinFax PRO 8.0 puede detactar los fallos en las transmisiones y realizar auto-correciones y auto-configuraciones, lo cual mejora su fiabilidad.

Para más información: http://www.symantec.com

NOVEDADES

Se presenta Microsoft Internet Explorer Suite 4.0

Durante el mes de abril, esta compañía ha anunciado la versión preliminar de Microsoft Internet Explorer 4.0, una versión de evaluación dirigida a los desarrolladores de software y de contenidos para Internet, así como a los usuarios.
Entre sus componentes más importantes destacan el correo electrónico y capacidades de exploración y otras para compartir aplicaciones y permitir la publicación y distribución de información por la Red, facilitando la integración de la Web. También incluye Microsoft NetMeeting 2.0, que permite mantener videoconferencias y charlas con diferentes personas dentro de Internet.
Otra de las novedades más interesantes para los programadores consiste en la inclusión del HTML Dinámico, que posibilita nuevas formas de interactividad en documentos HTML.
Cuando Microsoft consiga hacer de esta beta preliminar consistente y libre de errores, constituirá una solución perfecta para el usuario interesado en tener como herramienta de trabajo o de diversión a Internet.

LIBROS

Programando con Java

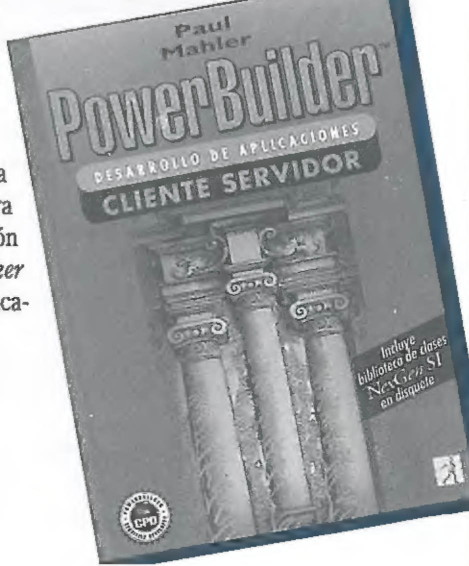
Java proporciona una forma completamente nueva de pensar la informática distribuida y un entorno de desarrollo verdaderamente transportable, sólido, dinámico, seguro y de alto rendimiento para transportar contenidos a través de Internet. Este lenguaje libera los computadores clientes de la Red de la dependencia respecto a los computadores principales para la ejecución de los contendios dinámicos.
Ante tal hito, la editorial PRENTICE HALL vuelve a sorprendernos con una nueva obra que, estructurada en 14 capítulos, desglosa esta tecnología conocida como la segunda revolución de Internet: el lenguaje de programación Java.
Editorial: PRENTICE HALL
Autor: Tim Ritchey
291 páginas
Idioma: Español
Incluye CD-ROM



PowerBuilder, aplicaciones cliente servidor

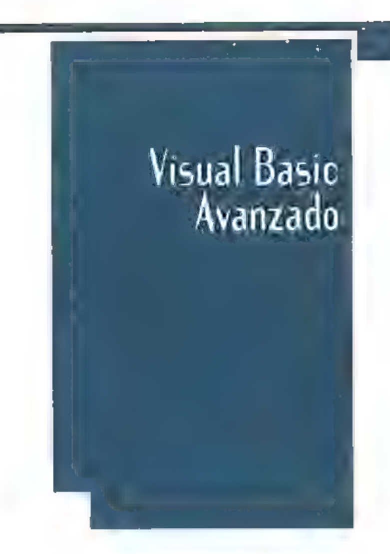
La versión PowerBuilder 4.0 destaca por su funcionalidad, al incluir nuevas características como soporte multiplataforma, mejoras del modelo orientado a objeto, del acceso a bases de datos, en la conectividad o la nueva tecnología de compilación. Esta versión incluye nuevas funcionalidades para los objetos, entre ellas las que permiten crear o ampliar las clases no visuales. También ha mejorado el acceso a objetos y clases externas, con herramientas DLL.

Además, PowerBuilder 4.0 soporta la anidación de informes y la arquitectura OLE 2.0 de Microsoft, la nueva versión OMG y *Common Object Request Broker* (CORBA), que se utilizan para la especificación de objetos.
Editorial: PRENTICE HALL
Autor: Paul Mahler
410 páginas
Idioma: Castellano



El control contenedor OLE

Juan Manuel Menéndez



En el artículo anterior se vio cómo realizar una aplicación contenedora OLE en Visual Basic sin codificación, jugando con los valores de las propiedades hasta conseguir el efecto deseado. En este número veremos cómo aumentar nuestro control utilizando los métodos y propiedades de los contenedores OLE.

Como primer paso tenemos que aclarar que los métodos y propiedades a los que vamos a tener acceso son los pertenecientes al contenedor, nunca los pertenecientes al objeto que pueda haber en el mismo. El acceso a los miembros de un objeto se realiza por medio de OLE automatización como veremos más adelante.

Objetos Contenedores OLE

En Visual Basic existe un Custom Control que da soporte a la inclusión de contenedores OLE en nuestras aplicaciones.

Este Custom Control nos permite, a través de sus métodos y propiedades (ver tablas 1 y 2 donde se muestran las más comunes), interaccionar con servidores OLE.

TABLA 1. Propiedades específicas del control contenedor OLE.

Close (Método control contenedor OLE)
Copy
CreateEmbed
CreateLink
Delete
DoVerb
Drag
FetchVerbs
InsertObjDlg
Move
Paste
PasteSpecialDlg
ReadFromFile
Refresh
SaveToFile
SaveToOle1File
Update

Con este Custom Control se tiene cierta capacidad de acción sobre el objeto:

- Clase de objeto.
- Tipo de objeto (enlazado o incrustado).

En la jerarquía de menos a más vamos a ver la posibilidad de tener el control sobre un servidor OLE, sin poseer conocimiento específico del mismo. Para ello, entraremos en un Custom Control de VB que nos facilitará este trabajo: el control Contenedor OLE.

TABLA 2. Métodos específicos del control contenedor OLE.

Close (Método control contenedor OLE)
Copy
CreateEmbed
CreateLink
Delete
DoVerb
Drag
FetchVerbs
InsertObjDlg
Move
Paste
PasteSpecialDlg
ReadFromFile
Refresh
SaveToFile
SaveToOle1File
Update

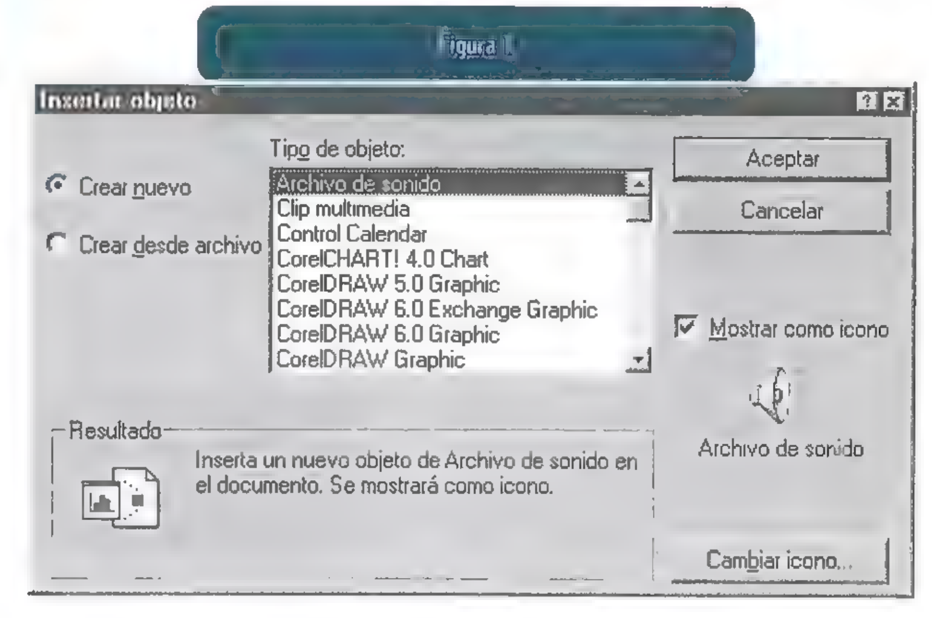
- Verbos que da al interfaz.
- Apariencia (como contenido o icono).

Trabajando con contenedores OLE

La primera propiedad que es necesario definir es la propiedad Class, que sirve para buscar en el Registry la entrada que identifica el servidor OLE y a partir de esta entrada obtener el resto de la información, tal y como se vio el mes anterior. Esta propiedad del objeto está disponible tanto en tiempo de diseño como en tiempo de ejecución.

Además de las formas normales de asignación de valores (ventana de propiedades, asignación de valores en ejecución), disponemos del cuadro de diálogo Insertar Objeto (ver figura 1), donde podemos ver los diversos servidores OLE disponibles en nuestro sistema.

Figura 1.



Cuando creamos un contenedor OLE en nuestro formulario, este cuadro de diálogo se activa permitiendo seleccionar el tipo de servidor que vamos a utilizar si se desea incluir esta información en tiempo de diseño. Si la selección se va a realizar en tiempo de ejecución, con el botón de cancelar cerramos el cuadro de diálogo sin haber realizado selección alguna. También se puede realizar esta selección actuando en la ventana de propiedades, como se hace con el resto de los Custom Control de VB.

Hay una pequeña diferencia de formato en los dos modos de seleccionar el tipo de servidor. En la ventana de propiedades aparece la clave del servidor tal y como está en HKEY_CLASSES_ROOT mientras en el cuadro de diálogo el servidor aparece el contenido de la carpeta. Centrándonos en el servidor de Word podemos comprobar que los valores que aparecen son:

Microsoft Word (Document).

Word.Document.6.

El cuadro de diálogo Insertar Objeto siempre está disponible seleccionando la opción Insertar Objeto del menú contextual del objeto del control. Si se selecciona en el cuadro de diálogo la opción Crear desde Archivo se abre otro cuadro de diálogo para seleccionar el archivo y un control Check Box nos permite indicar si queremos que el objeto sea incrustado o vinculado. Como se puede ver, si creamos desde un archivo del objeto (bien sea vinculado o incrustado), no indicamos qué tipo de servidor vamos a utilizar, ya que la extensión del nombre del fichero también está en el Registry.

Se pueden observar los diferentes menús de contexto que puede tener un control contenedor según el servidor que se haya escogido.

La definición de la clase de servidor en tiempo de diseño es bastante rígida, pero en tiempo de ejecución se pueden realizar las mismas operaciones para conseguir una mayor flexibilidad en nuestras aplicaciones.

El cuadro de diálogo se puede invocar con el método InsertObjDlg:

nombreControl.InsertObjDlg

El cuadro de diálogo es presentado de forma modal y su forma de operar es idéntica a la realizada en tiempo de diseño.

La otra forma de modificar la clase de objeto en tiempo de diseño se realiza

asignando el valor a la propiedad directamente:

nombreControl.Class = "Word.Document.6"

Otras propiedades con las que podemos controlar el modo de interacción con el usuario o ver el estado del servidor son:

AutoActivate. Con esta propiedad indicamos la forma de activar el objeto con tres formas posibles:

- Doble click.

- Al obtener el enfoque.

- Programáticamente.

AutoVerbMenu. Con esta propiedad indicamos si se desea que se despliegue un menú de contexto al pulsar el botón derecho del ratón.

AppIsRunning. Esta propiedad indica si la aplicación que ha creado el objeto está activa.

OLETypeAllowed tipo de objeto permitido:

- Incrustado.

- Enlazado.

- Cualquiera.

OLEType estado del objeto.

Hablando el mismo idioma

Las propiedades hasta ahora vistas no permiten más que definir sus características, sin poder ejecutar ninguna acción.

Por otro lado se ha dicho que los contenedores OLE permiten ejecutar acciones sobre un servidor OLE sin necesidad de conocer sus métodos ni propiedades. Esto que se presenta como antagónico, se ha solucionado de dos formas, según el problema:

Creación de API común para todos los servidores.

El servidor informa a los clientes de cuáles son sus métodos.

Los contenedores OLE funcionan con la segunda opción, en la que los servidores suministran información sobre las acciones que es posible ejecutar con ellos.

Estas acciones que se pueden realizar sobre un objeto en un contenedor OLE se denominan verbos.

El protocolo establecido sobre un objeto para obtener información acerca de los verbos es el siguiente:

El cliente obtiene la lista de verbos aceptados por el servidor.

El cliente invoca el verbo adecuado.

La obtención de los verbos se realiza en los siguientes pasos:

Actualización de la lista: Con el método FetchVerbs el servidor suministra los verbos soportados.

Obtención de los verbos. Los verbos están en una matriz y se puede obtener información sobre éstos con las propiedades:

- ObjectVerbsCount: Número de verbos disponibles.

- ObjectVerbs (n) : Elemento enésimo de la matriz de verbos. Éstos empiezan desde el elemento 0 (valor defecto), cuyo contenido se repite posteriormente.

Invocación del verbo. Con el método doVerb (n).

Figura 2

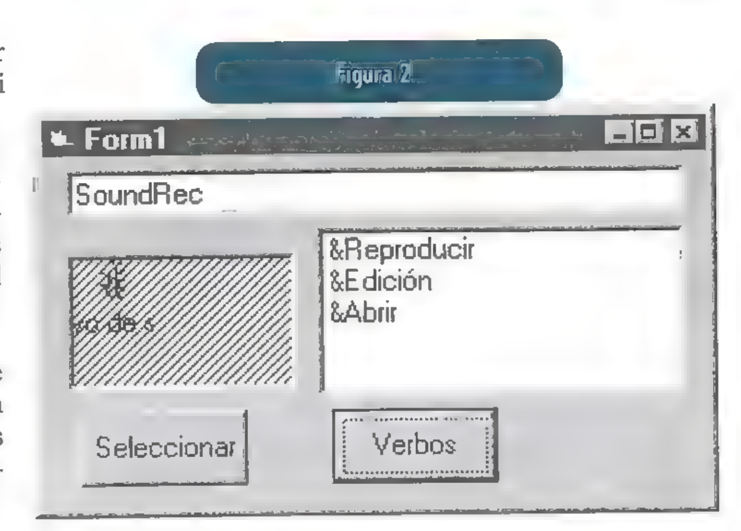
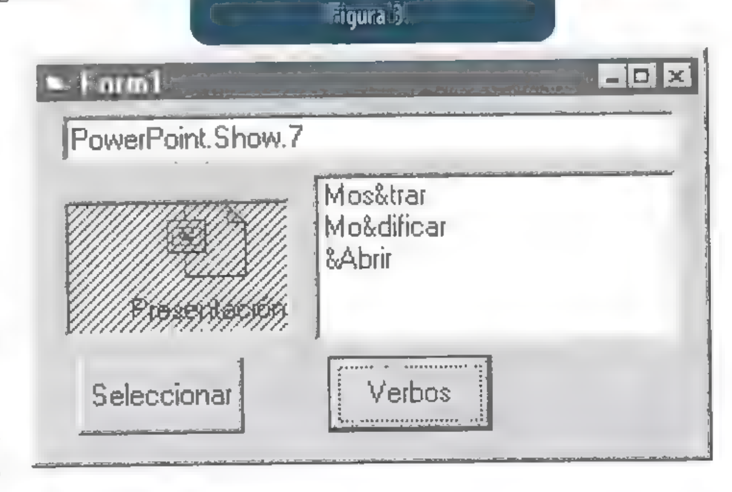


Figura 3



La figura 4 muestra el procedimiento que se utiliza para obtener todos los verbos.

En las figuras 2 y 3 podemos observar el resultado de la ejecución del procedimiento que aparece en la Figura. 4 para dos objetos distintos.

Una vez utilizado el objeto y para liberar memoria los objetos se liberan con el método Delete:

nombreObjeto.Delete

Trabajando con el portapapeles

Otro juego de propiedades y métodos del control nos permite trabajar con el portapapeles de Windows.

Para ello disponemos de los siguientes elementos:

- Propiedad PasteOk: si está a True (Verdadero), el contenedor puede capturar el contenido del portapapeles.
- Método Paste, que nos permite llevar el contenido del portapapeles a nuestro objeto.
- Método Copy, que lleva el contenido de nuestro objeto al portapapeles.
- Método PasteSpecialDlg, que nos presenta el cuadro de diálogo de pegado especial.

Creación de objetos

El cuadro de diálogo Insertar Objeto realiza tres tareas, como se puede observar al ejecutar el programa:

- Identificar la clase del objeto.
- Crear el objeto.
- Activar el objeto con el verbo por defecto.

En un programa más orientado a un usuario final, puede interesar tener un control mas fuerte del objeto con el fin de limitar el tipo de objeto que va a residir en el control, así como poder ejecutar el verbo deseado cuando sea necesario.

Los pasos a efectuar son los enumerados más arriba. La identificación del tipo de objeto se realiza asignando el valor requerido a la propiedad Class, como se ha visto anteriormente.

Luego es necesario crear el objeto. Para ello existen dos métodos, según el tipo de objeto que deseemos crear. Los métodos son:

CreateLink nombrefichero, [clase]

CreateEmbed nombreFichero,[clase]

El parámetro clase es opcional si el primer parámetro es el nombre de un fichero cuya extent de nombre esté en el Registry, algo equivalente a lo que ocurre en la opción Crear desde Archivo del cuadro Insertar Objeto. Si el primer parámetro es un string nulo se creará un objeto vacío, como ocurriría en Insertar Objeto sin marcar la opción Crear desde Archivo; en este caso es obligatorio indicar la clase de objeto en el segundo parámetro.

Una vez creado el objeto, éste no se activará hasta que no se le mande ejecutar un verbo. El método doVerb es el encargado de pasar al servidor la petición para ejecutar el verbo. Su sintaxis es:

nombreObjeto.doVerb(n)

Donde n puede tener los siguientes valores:

- > 0. Verbo que está en la posición n de la matriz de verbos del objeto.
- = 0. Verbo por defecto.
- < 0. Ejecución verbo estándar (ver figura 5)

En la figura 6 se puede ver el código para activar por programa el objeto.

FIGURA 4. Obtención e invocación de los verbos de un objeto.

```
Private Sub Command1_Click()
Dim i As Integer

OLE1.FetchVerbs
For i - 1 To OLE1.ObjectVerbsCount - 1
      List1.AddItem OLE1.ObjectVerbs(i)
   Next i
End Sub

Private Sub List1_DblClick()
On Error Resume Next

OLE1.DoVerb (List1.ListIndex)
End Sub
```

Salvando y recuperando ficheros

Existen unos métodos que nos permiten guardar el contenedor con el que estamos trabajando y posteriormente recuperarlo.

Estos métodos guardan el contenedor OLE, es decir, los datos e información en un formato propio no reconocible por el servidor, únicamente recuperable por el método propio del contenedor.

Hay dos métodos para guardar el contenedor:

- SaveTofile.
- SaveToOLE1File.

La diferencia entre ambos es el formato interno, pero funcionalmente son idénticos, pues se pueden recuperar con el método

FIGURA 5. Verbos estándar.

1 Activa el objeto para edición. Si la aplicación que ha creado el objeto acepta activación in situ, el objeto se activa dentro de el control contenedor OLE.

2 Abre el objeto en una ventana de la aplicación distinta. Si la aplicación que ha creado el objeto acepta activación in situ, el objeto se activa en su propia ventana.

3 En objetos incrustados, oculta la aplicación que ha creado el objeto.

4 Si el objeto acepta activación in situ, activa el objeto para activación in situ y muestra las herramientas de la interfaz de usuario. Si el objeto no acepta activación in situ, el objeto no se activa y se produce un error.

5 Si el objeto acepta activación in situ, activa el objeto para activación in situ y muestra las herramientas de la interfaz de usuario. Si el objeto no acepta activación in situ, el objeto no se activa y se produce un error.

6 Si el objeto acepta activación in situ, activa el objeto para activación in situ y muestra las herramientas de la interfaz de usuario. Si el objeto no acepta activación in situ, el objeto no se activa y se produce un error.

FIGURA 6. Diversos modos de activación de un objeto.

```
Private Sub Command3_Click()
If OLE1.AppIsRunning = True Then
   OLE1.Delete
End If
If menuInsertar.Checked = True Then
   OLE1.InsertObjDlg
Else
   OLE1.Class = "PowerPoint.Show.7"
   OLE1.CreateEmbed "", OLE1.Class
   OLE1.DoVerb (0)
End If
Text1.Text = OLE1.Class
End Sub
```

ReadFromFile

Sin indicar el tipo de formato.

Junto al programa de demostración, aparecen tres ficheros, el fichero con la extensión wav es un fichero de origen, los ficheros prueba00.OLE y prueba10.OLE han sido salvados con los métodos SaveToFile y SaveToOLE1File respectivamente, se puede observar la diferencia de tamaño entre ellos y con respecto al original. Si se intenta ejecutar con el programa de sonido cualquiera de los ficheros .OLE, se observa que el fichero no es reconocido como un fichero WAV, en cambio si lo abrimos con el programa, observamos cómo el tipo de fichero es reconocido al aparecer sobre el contenedor el símbolo del programa asociado en el Registry.

RECAPITULANDO

Por medio del Control Contenedor OLE hemos realizado las siguientes etapas en el control de un objeto OLE:

Definición definitiva del objeto en tiempo de diseño.

Gestión del control en tiempo de ejecución:

a) Sin capacidad de gestión: Propiedades AutoActivate, AutoVerbMenu.

b) Poca capacidad de gestión: Método InsertObjDlg.

c) Gran capacidad de gestión: Métodos Create [Linked|Embed], doVerb, Propiedades: ObjectVerb, ObjectVerbCount

Presentación: Icono, Contenido.

Tipo de Objeto: Todos los definidos en el Registry (Word, Excel, Word Perfect...)

- Enlazados o incrustados.

- Nuevos o partiendo de ficheros existentes.

En el próximo número se verá otra forma de trabajar con los servidores OLE, a través de los métodos y propiedades específicos, con lo que conseguiremos un control total sobre los servidores.



El modelo de objetos de JavaScript

Alejandro M. Reyero Abad
axl@las.es

El modelo de objetos de JavaScript es sencillo, a la par que potente. Veremos a continuación cómo se utilizan los objetos predefinidos en el cliente (browser HTML), las propiedades de los mismos y aprenderemos a definir y utilizar nuestros propios objetos JavaScript.

Un objeto JavaScript es un constructor con propiedades que son variables JavaScript, además estas mismas variables pueden, a su vez, ser otros objetos.

Las variables se usan para almacenar valores (lógicamente) y éstos pueden ser de tres tipos:

- Números: No necesitan mayor explicación. 5 ó 3.14159 son algunos ejemplos.

- Cadenas: Siempre van entre comillas. Un ejemplo es "En un lugar de...".

- Booleanos: Pueden ser true o false. Para definir un booleano lo declaramos con new Boolean(). Lo que pongamos entre los paréntesis determinará el valor del booleano. Devuelven true: new Boolean (true), new Boolean("false") y new Boolean("Viva Yo"). Como podemos apreciar, todos los strings no vacíos devuelven true (incluso el string "false"). Éstos devolverían false: new Boolean(false), new Boolean(""), new Boolean(), new Boolean(0) y new Boolean(null).

En las variables podemos almacenar valores y manipularlos.

La página asigna.htm es un ejemplo de uso de variables y asignación de valores a las mismas.

asigna.htm

```
<script>
a = new Boolean("False");
b = 5;
c = 3.14159;
d = "En un lugar de...";
document.write("a = " + a + "<br>");
document.write("b = " + b + "<br>");
document.write("c = " + c + "<br>");
document.write("d = " + d + "<br>");
</script>
```

Aparece en la misma un elemento que aún no conocemos: el método document.write. Su utilidad es obvia: escribir algo en el documento. No obstante lo veremos con detenimiento más adelante.

Primeros objetos

El browser HTML (Navigator 2.0+ o Internet Explorer 3.0+ son los únicos, hasta ahora, que incorporan JavaScript) incluye una serie de objetos predefinidos, a los que podemos añadir nuestros objetos con sus propiedades correspondientes.

Antes de comenzar con objetos y propiedades, conviene recordar que, al igual que Java, JavaScript es un lenguaje *case sensitive*, así que tanto los nombres de los objetos como los nombres de las pro-

piedades distinguen entre mayúsculas y minúsculas.

Para acceder a las propiedades de un objeto JavaScript usaremos el siguiente esquema:

```
nombreDelObjeto.nombreDeLaPropiedad
```

Para definir una propiedad le asignaremos un valor. Por ejemplo, si tenemos un objeto llamado ganador (veremos más tarde cómo crear nuestros propios objetos) le asignaríamos propiedades de la siguiente manera:

```
ganador.temporada = "97 98"
ganador.equipo = "Sporting"
ganador.fundacion = 1908
```

Este tipo de array tiene cada elemento índice asociado con el valor de una cadena. Veremos cómo funciona creando una función genérica que reciba como parámetro un objeto y muestre las propiedades de ese objeto con sus valores correspondientes:

```
Necesita: Un objeto.
Produce:    Una cadena (resultado) con las propiedades
// del objeto y su valor, preformateado para su
  impresión.
function muestra(objeto) {    var resultado = ""
  for (var z in objeto)
    resultado = resultado + z + " = " + objeto[z] + "\n"
  return resultado;
}
```

Si llamamos a la función con muestra(ganador) el resultado sería:

```
temporada = 97 98
equipo = Sporting
fundacion = 1908
```

Objetos predefinidos por el browser

Nuestro browser crea una serie de objetos cada vez que cargamos una página, correspondiéndose éstos con los contenidos de la misma.

En todas las páginas encontraremos, al menos, los siguientes objetos:

Navigator: contiene propiedades con el nombre y la versión del browser que estamos utilizando, los tipos MIME soportados por el cliente y los plug-ins que el cliente tiene instalados.

Window: contiene las propiedades referentes a toda la ventana o a la ventana hija en un documento con frames.

Document: agrupa las propiedades del documento actual, como el título, los formularios, color de fondo, los links. etc.

History: contiene las URLs que el usuario ha visitado anteriormente (desde la version 2.01 de Navigator esto se ha limitado bastante por razones de seguridad).

Location: propiedades del URL actual.

Conviene no confundir el objeto window con el objeto document, window es el objeto "padre" que contiene en su interior al objeto document. Estará mas claro cuando veamos la jerarquía de objetos de JavaScript.

Tomemos como ejemplo la página a la que hemos llamado ejemprop.htm.

Como todos los documentos, ejemplo.htm contiene los objetos navigator, window, document, history y location, siendo algunos de los valores "estándar" de sus propiedades los siguientes:

```
document.title = "Ejemplo de Propiedades de una página Web"
document.bgColor = #888888
document.fgColor = #000000
location.href = http: www1.las.es ~axl/js/ejemplo.htm
location.host = www1.las.es
history.length = 3
```

Encontraremos en todas las páginas los objetos window, document, location e history

Teniendo en cuenta el código fuente de la página el browser creará los objetos document.primerForm, document.primer Form.Text1 y document.primerForm. Boton con las siguientes propiedades:

ejemprop.htm

```
<HEAD>
<TITLE>Ejemplo de Propiedades de una
p&aacute;gina Web</TITLE>
</HEAD>
<BODY bgcolor=Nº888888
fgcolor=Nº000000>

<FORM NAME="primerForm" ACTION="
algoharemos()"
METHOD="get">
Esto es un TextBox:
<INPUT TYPE="text" NAME="Text1" SIZE=15
VALUE="Esto Mismo">
<INPUT TYPE="button" NAME="Boton"
Value="Para que Aprietes"
onClick="actualiza(this.form)">
</FORM>

</BODY>
```

```
document.primerForm.action = algoHaremos()
document.primerForm.method = get
document.primerForm.Text1.name = Text1
document.primerForm.Text1.value = "Esto Mismo"
document.primerForm.Boton.name = Boton
document.primerForm.Boton.value = "Para que Aprietes"
```

Todas las referencias a las propiedades comienzan por document seguidas por el nombre del formulario y finalmente el de la propiedad (o el nombre del elemento seguido del de la propiedad del elemento).

Para comprender el porqué de esta nomenclatura veremos en qué consiste la jerarquía de objetos de Navigator (definida por Netscape).

Jerarquía de objetos de Navigator

La estructura de objetos de JavaScript es jerárquica y sigue la estructura de una página HTML.

En esta jerarquía, los "descendientes" de un objeto son, a su vez, propiedades del objeto. En el ejemplo anterior, el formulario llamado primerForm era un objeto, pero también era una propiedad del objeto document, y podíamos referirnos a ella con document.primerForm. El esquema "jerarca" ilustra la Jerarquía de objetos de Navigator.

Para referirse a alguna propiedad de estos objetos es necesario especificar el nombre del objeto y todos sus "antepasados", con la única excepción del objeto window (no es necesario incluirlo en la referencia ya que es el objeto "padre" del resto de los objetos en un documento HTML).

Usando objetos predefinidos

Objeto form:

Para cada formulario en una página HTML, el browser crea un objeto form al que podemos darle nombre con el atributo NAME:

```
<FORM NAME="elNombre">
<INPUT TYPE="checkbox" NAME="chequea"
CHECKED>
.....
</FORM>
```

A partir de este momento tenemos un objeto llamado elNombre que toma sus propiedades de este formulario (por ejemplo document.elNombre.chequea).

Todos los formularios de un documento se almacenan en un array llamado forms, teniendo el primer formulario de una página la posición 0 en el array, el segundo la 1 y así sucesivamente. Por tanto podremos referirnos a ellas con forms[0], forms[1], etc.

Esto implica que los formularios no tienen que ser nombrados obligatoriamente para acceder a ellos (siempre que sepamos la posición que ocupan en la página HTML). Para el ejemplo anterior, si elNombre fuera el tercer formulario en un documento, nos referiríamos a su propiedad "chequea" como document.forms [2].chequea.

A su vez, los elementos dentro de un formulario, como cajas de texto, checkboxes o botones, son almacenados en un array llamado elements, que funciona exactamente igual.

Para conocer todos los arrays disponibles a partir de la versión 3.0 de Netscape Navigator véase la tabla llamada arrays.

Objeto document:

Todas las páginas contienen un, y sólo un, objeto document, siendo el "padre" de todos los objetos form, anchor y link que contenga la página.

El objeto document es especialmente útil, ya que puede generar código HTML con los métodos write y writeln (la diferencia entre ambos es que el segundo añade un fin de línea). Por ejemplo:

```
document.write("<H1>Esto sale en letra gruesa</H1>")
```

El objeto document también tiene los eventos onLoad y onUnload, que nos permiten ejecutar código JavaScript cuando el usuario carga o descarga nuestra página. Lo vemos mejor con un ejemplo:

```
<HEAD>
<SCRIPT>
function alCargar()
{
document.writeln("<H1>Esto sale en letra gruesa
</H1>");
}
</SCRIPT>
</HEAD>
<BODY onLoad="alCargar()">
</BODY>
```

Objeto location:

El objeto location contiene las propiedades basadas en el URL actual. Por ejemplo, la propiedad hostname es el dominio del server en el que se encuentra ubicada la página que estamos visitando.

El objeto location tiene dos métodos:

- replace, que carga el URL actual sobre la última entrada en el history.
- reload, que hace que la ventana actual se vuelva a cargar.

Objeto history:

Contiene una lista de cadenas con los URLs que el cliente (browser) ha visitado. Se puede acceder a las entradas actual, siguiente y anteriores del history usando los objetos de history current, next y previous. Se puede acceder al resto de entradas con el array history (de la misma manera que el array form que ya hemos visto).

Todos los formularios del documento se almacenan en un array llamado forms

También se puede hacer que el cliente vaya a cualquier entrada en el history con el método go:

Cargará la tercera entrada a la anterior en la que nos encontramos, mientras que:

```
history.go(0)
```

cargará de nuevo la página actual.

Objeto Navigator:

Contiene información sobre la versión de nuestro browser, por ejemplo, con la propiedad appName conoceremos el nombre del mismo (Navigator, IExplorer...) y con appVersion conoceremos su versión.

Objeto window:

Tiene varios métodos dedicados a la creación de cajas de diálogo y nuevas ventanas.

Jerarquía de objetos de Navigator

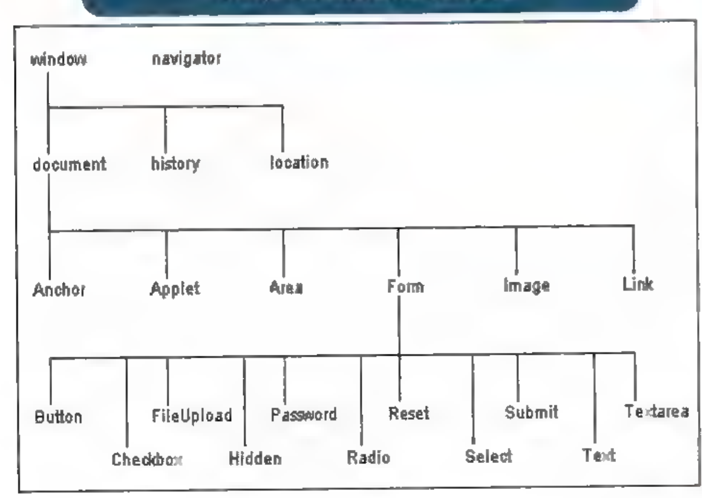


TABLA "ARRAYS"

Array	Descripción
anchors	Almacena todos los tags "A" que contengan el atributo NAME del documento.
applets	Almacena todos los tags "APPLET" del documento.
arguments	Almacena los argumentos pasados a una función.
elements	Almacena los elementos de un form.
embeds	Almacena los tags "EMBED" del documento.
forms	Almacena los forms del documento.
frames	Almacena todos los tags "FRAME" de una ventana que contengan el tag "FRAMESET".
history	Almacena las entradas en el history de una ventana.
images	Almacena los tags "IMG" del documento.
links	Almacena los tags "A HREF="..."" del documento así como los objetos Link creados con el método link.
mimeTypes	Contiene todos los tipos MIME que contiene un documento, tanto internos como los añadidos por los Plug-ins.
options	Contiene todos las opciones en un objeto Select (los tags "OPTION").
plugins	Contiene todos los Plug-ins instalados en el cliente.

Entre las cajas de diálogo nos encontramos con alert (abre una caja de alerta) y confirm (abre una caja de confirmación). Veamos un ejemplo:

```
<HEAD>
<script>
function cajaConfirma()
{
confirm("Vamos muy bien");
}
function cajaAlerta()
{
alert("Cuidadin");
}
</script>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" VALUE="Confirma" onClick="cajaConfirma()">
<INPUT TYPE="button" VALUE="Alerta" onClick="cajaAlerta()">
</FORM>
</BODY>
```

Como habíamos visto, no hace falta poner el objeto window para llamar a algún método del mismo o hacer referencia a alguna de sus propiedades. Los puristas tal vez prefieran sustituir confirm("Vamos muy bien") por window.confirm("Vamos muy bien") y alert("cuidadin") por window.alert("cuidadin"). Cuestión de gustos, aunque, teniendo en cuenta la "espectacu-

lar" velocidad de InfoVía, cuantos más bytes ahorremos en nuestras páginas, mejor.

Se pueden crear nuevas ventanas de navegación con el método open. El siguiente comando crearía una ventana llamada buenosJuegos que cargaría la página web de 3DRealms:

```
buenosJuegos=window.open("http. www3drealms.com")
```

Con el nombre buenosJuegos podremos referirnos a las propiedades, métodos y contenidos de la ventana, pero no podremos usar el nombre buenosJuegos para referirnos a esta ventana desde otra ventana, tendremos que darle un segundo nombre:

```
buenosJuegos=window.open("http. www.3drealms.com",
"buenosWindow")
```

Cuando abrimos una ventana, se pueden especificar atributos como el tamaño de la misma y si tiene un toolbar, scrollbar o campo location. La siguiente sentencia creará una ventana sin toolbar pero con scrollbar:

```
buenosJuegos=window.open("http:, www.3drealms.com",
"buenosWindow", "toolbar=no, scrollbar=yes")
```

Para cerrar una ventana usaremos el método close, teniendo en cuenta que no podemos cerrar un frame sin cerrar toda la ventana "padre".

Las siguientes sentencias cerrarían la ventana actual:

```
window.close()
self.close()
close()
```

y para cerrar la ventana buenosJuegos:

```
buenosJuegos.close()
```

La página que Raúl Alonso (de Láser Internet Center) tiene de Prince es un ejemplo estupendo de lo que puede llegar a hacer abriendo y cerrando nuevas ventanas (véase la parte "video" de su página http://www.las.es/TheArtist).

Como ampliación del objeto window veremos a continuación cómo se actualizan frames desde JavaScript.

Frames y JavaScript, introducción

Un frame es un tipo de ventana que permite mostrar varios "marcos" en la misma pantalla, cada uno de ellos con su URL correspondiente. Una serie de frames, o frameset, constituye una página.

Para crear un frame usaremos el tag HTML <FRAMESET> Ver ejemplo 1.

Este ejemplo crea una página con 6 frames, para verlo en "acción": http://www.inode.es/~axl

En el primer frameset le decimos al browser que nuestra página contiene 3 frames, que ocuparán 3 columnas en el texto, teniendo la primera y la última una anchura de 100 pixeles, estando el resto del espacio ocupado por el frame central.

El primer frame (recordemos que son independientes) contiene a su vez otros 4 frames, todos del mismo tamaño, ocupando cada uno un 25% de las líneas disponibles.

Conviene destacar la ausencia de bordes en los frames (frameborder="no" y border="no") y la inclusión de un texto para la gente que visita nuestra página con un browser que no soporta frames (tag <NOFRAME>).

A pesar de incluir dos framesets, todos los frames de la página tienen el mismo padre, ya que el padre de un frame es la ventana en la que se incluye, y un frame, no un frameset, define una ventana.

Para referirnos a los frames podemos usar el array frames, donde cada frame ocupa una posición:

```
izq1 es frames[0]
izq2 es frames[1]
izq3 es frames[2]
izq4 es frames[3]
centro es frames[4]
esquina2 es frames[5]
```

Con JavaScript podremos hacer que 2 o más frames se actualicen simultáneamente. Como ejemplo vamos a ver el código del frame izq1 (izq1.htm).

Vemos que es bastante sencillo, simplemente capturamos el evento onClick sobre la imagen y llamamos a la función cambiaFrame, la cual actualiza los frames 4 y 5 de parent (la ventana en la que él mismo se encuentra, o sea, su ventana padre).

Creando nuestros propios objetos

Para crear nuestros objetos tenemos que seguir dos pasos:

1. Definir el objeto con una función constructor.

2. Crear una instancia del objeto con new.

A la hora de definir nuestro objeto tenemos que crear una función que especifique su nombre, propiedades y métodos.

Por ejemplo, para definir el objeto ganador que habíamos utilizado al principio:

```
function ganador(temporada, equipo, fundacion) {
    this.temporada = temporada
    this.equipo = equipo
    this.fundacion = fundacion
}
```

TE BUSCAMOS...

SI TIENES AMPLIOS CONOCIMIENTOS O EXPERIENCIA EN ALGUNO DE ESTOS CAMPOS

- Programación en Visual Basic, Visual C++, Delphi, HTML, JAVA, JAVAScript, CGI
(Ref. PROG)
- Lenguajes multimedia como Director, Authorware, Toolbook
(Ref. MULT)
- Dominio de sistemas operativos DOS, Windows 95, Windows NT, UNIX, Linux, OS2.
(Ref. S.O)
- Redes, Conectividad y Sistemas abiertos
(Ref. RED)
- Diseño y autoedición en CorelDraw, Photoshop, FreeHand, QuarkXPress, Page Maker
(Ref. DIS)
- Infografía con 3D Studio, LightWave
(Ref. INF)
- Internet, Navegación, Videoconferencia, ActiveX, Plug-Ins, Shockwave, RDSI
(Ref. NET)
- Nuevas tecnologías, MMX, USB, DVD, Telefonía móvil
(Ref. TECN)
- Instalación, reparación y mantenimiento de software y hardware
(Ref. INST)
- Hardware: Tarjetas gráficas, Modems, Cámaras digitales, Scaners, Tarjetas de sonido
(Ref. HARD)
- Inteligencia Artificial, Sistemas Expertos, Redes neuronales, Robótica
(Ref. SE)

COLABORA CON NOSOTROS

Envíanos un curriculum vitae con la referencia correspondiente a la siguiente dirección:
TOWER COMMUNICATIONS - APARTADO DE CORREOS 54.283 - 28080 MADRID

EJEMPLO 1. Tag FRAMESET

```
<FRAMESET COLS="100,*,100" frameborder="no" border="0">
<FRAMESET ROWS="25%,25%,25%,25%" frameborder="no" border="0">
<FRAME SCROLLING="no" name="izq1" NORESIZE SRC="izq1.htm">
<FRAME SCROLLING="no" name="izq2" NORESIZE SRC="izq2.htm">
<FRAME SCROLLING="no" name="izq3" NORESIZE SRC="izq3.htm">
<FRAME SCROLLING="no" name="izq4" NORESIZE SRC="izq4.htm">
</FRAMESET>

<FRAME SCROLLING="yes" NAME="centro"  NORESIZE SRC="centro.htm">
<FRAME SCROLLING="no" NAME="esquina2" NORESIZE SRC="esquina2.htm">

<NOFRAME>
Lo siento, esta p&aacute;gina contiene frames y tu browser no los
soporta.
</NOFRAME>
</FRAMESET>
```

izq1.htm

```
<HEAD>
<LINK REV="made" HREF="mailto:axl@las.es">
<SCRIPT LANGUAJE="JavaScript">
<!-- Para esconderlo del Lynx

function cambiaFrame() {
  parent.frames[4].location = 'america.htm'
  parent.frames[5].location = 'derame.htm'
}
// Aqui se acabo -->
</SCRIPT>
</HEAD>
<BODY bgcolor="black">
<CENTER>
<A HREF="Nº" onClick="cambiaFrame()">
<IMG SRC="imagenes/america.gif" border=0 ALT=" Buscadores Americanos >
width=70 height=90"</A>
</CENTER>

"/BODY"
```

Con this asignamos valores a las propiedades de los objetos basándonos en los valores pasados a la función.

Para crear un objeto llamado queViene:

```
queViene = new ganador("97/98", "Sporting", 1908)
```

Ahora el valor de queViene.temporada es igual a "97/98", queViene.equipo es igual a "Sporting", etc.

Como habíamos visto, un objeto puede tener propiedades que son a su vez otros objetos, por ejemplo, podríamos crear un objeto llamado plantilla, con los jugadores del equipo ganador, y, redefiniendo el objeto ganador para que admita una nueva propiedad, crear nuevos objetos ganador que incluyan el objeto plantilla.

Para añadir propiedades a objetos que ya hemos definido usaremos la propiedad prototipe, que define una propiedad que será compartida por todos los objetos de un mismo tipo, en vez de por una sola instancia del objeto. Como ejemplo, vamos a añadir la propiedad estadio:

```
ganador.prototype.estadio = null
queViene.estadio = "El Molinón"
```

Conclusiones

Hemos estudiado el modelo de objetos de JavaScript, cómo funcionan las propiedades de los mismos, los métodos más importantes de los objetos predefinidos, y la creación de nuestros propios objetos. En futuros artículos profundizaremos en el tema de los frames y su relación con JavaScript, añadiremos nuevos métodos a nuestros objetos, conoceremos más objetos predefinidos, veremos cómo conectar Java Applets con JavaScript mediante LiveConnect (usando los APIs netscape.javascript.* y netscape.plugin.*) y otros temas avanzados (JavaScript URLs, cookies, detectar plug-ins...).

Estructuras de control y otras lindezas

Oscar Prieto Blanco



Afortunadamente un lenguaje de programación es menos complicado de manejar que la vida misma. En el artículo de este mes nos introduciremos en la manera que tienen los programas de Java de manipular su mundo. Estudiaremos las estructuras que nos permitirán controlar el flujo de ejecución de un programa, estas construcciones son similares a las existentes en el lenguaje C, por lo que nos detendremos poco tiempo con ellas. Para poder trabajar a gusto con estas sentencias de control, tendremos que echar previamente un vistazo a los distintos tipos de operadores con los que cuenta el lenguaje, también muy influidos en su sintaxis por el lenguaje C.

Pero antes de nada debemos revisar el concepto de array (también llamados arreglos o matrices) en Java.

Arrays de objetos

Un array es simplemente una secuencia de objetos, todos del mismo tipo y reunidos juntos bajo un mismo nombre identificador.

Los arrays de objetos son simplemente arrays de handles

Los arrays son considerados objetos en Java; no solamente simples punteros a memoria como en C. Un array en Java tiene garantizada su inicialización y no puede ser accedido fuera de su rango (fuente de problemas en C que sí lo permite). Cuando creamos un array de objetos realmente estamos construyendo un array de handles y cada uno de esos handles es automáticamente inicializado a null. Nosotros debemos asignar un objeto a cada handle antes de usarlo y, si intentamos usar un handle que todavía es null, seremos informados en tiempo de ejecución.

También podemos crear un array de tipos básicos. De nuevo el compilador garantiza su correcta inicialización, ya que fuerza a que asignemos valores iniciales a dichos tipos.

Para crear un array en Java son necesarios tres pasos:

1) Declarar un handle para contener al array. Ejemplo:

```
String Libro[];
// también permitido
String[] Paginas;
```

2) Crear un nuevo objeto array y asignarlo al handle. Para lograr esto tenemos dos caminos:

* Mediante el uso de new. Ejemplo:

Al sostener que existen dioses, ¿no será que nos engañamos con mentiras y sueños irreales, siendo que sólo el azar y el cambio mismo controlan el mundo?

```
// Creo un array de 5 elementos
String[] s = new String[5];
```

Cuando creamos un array de objetos usando new, todos sus elementos son inicializados para nosotros: 0 para matrices numéricas, false para boolean, '\0' para caracteres y null para objetos.

* Directamente inicializando los contenidos del array. Ejemplos:

```
// Arrays de 3 elementos.
// De tipos básicos:
int a1[] = { 1, 2, 3};
// De objetos:
Integer[] a2 = {
new Integer(1),
new Integer(2),
new Integer(3),
    };
```

3) Asignar objetos a cada elemento particular del array. En este punto debemos mencionar que cada objeto matriz contiene una variable, llamada length, que almacena el número de elementos de dicho array. Para acceder a un elemento concreto del arreglo utilizaremos un índice cuyo valor varía entre 0 y la longitud del array menos uno. Debemos recordar que Java no permite acceder a elementos del array con un índice mayor que (length - 1) lanzando una excepción en caso de que el programador lo intente.

Ejemplos:

```
// Calculo nº de elementos:
int len = a1.length;
// Acceso a elmts. particulares:
int elem = a1[0];
// No permitido en Java:
int elm = a1[3];
```

Java no soporta arrays multidimensionales. Sin embargo, pueden ser simulados mediante el uso de arrays de arrays, los cuales pueden ser inicializados de múltiples formas. Ejemplos:

```
//////////////////////////
// Primera manera //
//////////////////////////
int k[][] = new int[5][4];
// Asignación individual
k[1][3] = 999;
//////////////////////////
// Segunda manera //
//////////////////////////
// Primero declaramos el handle
int z[][];
int outerSize = 5;
int innerSize = 4;
// Después asignamos el tamaño
z = new int[outerSize][innerSize];
//////////////////////////
// Tercera manera //
//////////////////////////
// Sólo es necesario conocer una
// dim. para crear un obj. array
int j[][] = new int[2][];
j[0] = new int[4];
j[1] = new int[4];
//////////////////////////
////// Cuarta manera //////
//////////////////////////
// La más directa:
int a1[][] = {
    { 1, 2, 3,},
    { 5, 6, 7,},
};
```

Operadores en Java

Un operador toma uno o más argumentos y produce un nuevo valor. Adicionalmente un operador puede cambiar el valor de un operando, aspecto que es ampliamente utilizado. Dado que la utilización y sintaxis de los operadores son similares a las del lenguaje C, y por otro lado, bastante obvia, en este apartado comentaré tan sólo unos cuantos hechos puntuales. Java soporta operaciones aritméticas, varias formas de asignación, incremento y decremento, operaciones lógicas y puede tratar con los datos a nivel de bits (bitwise operators). En la tabla 1 podemos ver un listado de todos ellos organizados por categorías.

Fijándonos en la tabla podemos ver que aparece un operador que no existía en C, el operador >>>. Este símbolo representa un desplazamiento hacia la derecha (pero rellenando con ceros, en oposición con >> que desplaza a la derecha pero conserva el bit de signo en el relleno) necesario por la ausencia de tipos sin signo en el lenguaje.

Precedencia de operadores

La precedencia de operadores define cómo una expresión es evaluada cuando varios operadores están presentes. Java tiene reglas específicas que determinan el orden de evaluación. La más sencilla de memorizar es que la multiplicación y la división ocurren antes que la adición y la sustracción. Aunque existen numerosas reglas, lo único que debemos recordar es que si tenemos dudas, lo más cómodo es usar paréntesis.

Control del flujo de un programa

Las sentencias de control constituyen uno de los pilares de cualquier lenguaje, ya que en ellas se apoya la estructura de los programas. Java utiliza la misma sintaxis que el C para las construcciones de control de ejecución de un programa. Por lo que la mayoría de los tópicos comentados en este apartado resultarán familiares a los lectores conocedores de este lenguaje.

En Java las sentencias de control de flujo se pueden dividir en dos grupos: las de toma de decisiones (bifurcación del control), simples o múltiples, y las de control de bucles o iteraciones.

Bifurcación del control.

TABLA 1. Clasificación de operadores.

<table>
<tr><th>Categoría</th><th>Operadores</th></tr>
<tr><td>Aritméticos</td><td>+ - * / %</td></tr>
<tr><td>Comparación y lógicos</td><td>< > >= <= == != && ! ||</td></tr>
<tr><td>Operadores bit a bit</td><td>& | ^ << >> >>> ~ &= |= ^=</td></tr>
<tr><td>Asignaciones</td><td>= += -= /= %=</td></tr>
<tr><td>Asignación a nivel de bit</td><td>& | ^ <<= >>= >>> ~ &=|=^=</td></tr>
<tr><td>Operador ternario</td><td>?:</td></tr>
<tr><td>Incremento</td><td>++</td></tr>
<tr><td>Decremento</td><td>--</td></tr>
</table>

Dependiendo del resultado de una condición ejecutamos un grupo de sentencias u otro. Sintaxis:

```
if(condición){
      sentencias
}
// El else es opcional
else{
sentencias
}
```

Donde condición debe ser obligatoriamente una expresión booleana, es decir, una expresión que retorne true o false (a diferencia del C que permite utilizar también una expresión aritmética). Por ejemplo:

```
int i = 4, j = 6;
boolean b = i<j;
// el valor de b es true
```

Java no permite el uso de expresiones aritméticas para evaluar condiciones

El operador ternario suele ser usado para evaluar condiciones sencillas, su estructura es:

```
Condición?sentencia1:sentencia2
```

Si el resultado de condición es true se ejecuta sentencia1 y si es false será ejecutada la sentencia2. Este operador también puede utilizarse para devolver un valor, como podemos ver en el siguiente ejemplo, donde N toma un valor u otro dependiendo de la condición:

```
int i = 4, j=6;
int N = i>j?0:1;
// N contiene un 1
```

Cuando una decisión es múltiple y compleja, comprobando distintos valores de un mismo identificador, es posible utilizar otra estructura alternativa para simplificar el proceso, llamada de bifurcación múltiple. Su sintaxis es:

```
switch(selector-entero){
case valor-entero1:
Sentencias;
break;
case valor-entero2:
Sentencias;
break;
// ... Otros case ...
// Esta etiqueta es opcional
default:
Sentencias;
}
```

Selector-entero es una expresión que produce un valor entero. El switch compara el valor del selector-entero con cada valor-entero. Si encuentra una coincidencia se ejecutan las correspondientes sentencias. Si no existe coincidencia se ejecutarán las sentencias de la etiqueta default. La palabra reservada break es opcional, si no la colocáramos en alguna etiqueta case el código de las siguientes sentencias case se ejecutaría hasta que fuera encontrado un break.

En Java no contamos con tipos aritméticos sin signo

Control de bucles

En Java existen tres estructuras distintas para la programación de bucles:

1) Bucle condicional con comprobación previa. Sintaxis:

```
while(condición){
    Sentencias
}
```

Si la condición inicial no es cierta nunca, no se producirá ninguna iteración. Ejemplo:

```
public class WhileTest{
  public static void
  main(String args[]){
double r = 0;
while(r<0.50d){
  // random genera un valor
  // entre 0 y1
  r = Math.random();
        System.out.prinln(r);
}
                    }
}
```

2) Bucle condicional con comprobación posterior. Sintaxis:

```
do{
 Sentencias
}while(condición);
```

El funcionamiento es análogo al de while, con la única diferencia del momento en que se realiza la evaluación de la condición. Un bucle de este tipo se ejecuta siempre al menos una vez porque primero se entra en el bucle y al final se comprueba.

3) Bucle condicional con inicialización y actualización. Sintaxis:

```
for(Inicializacion;Condicion;Paso){
Sentencias;
}
```

Un bucle for realiza una inicialización antes de la primera iteración. Después realiza un test de la condición y, al final de cada iteración, realiza algún tipo de actualización. Ejemplo:

```
public class Arrays{
  public static void
  main(String args[])
  {
      int a1[] = {1,2,3,4};
      int a2[];
      a2 = a1;
for(int i=0;i<a2.length;i++)
{     // i solo tiene alcance
      // dentro de este bloque
a2[i]++;
}
for(int i=0;i<a1.length;i++)
      prt("a1[" + i + "] = " +
        a1[i]);
  }
  static void prt(String s){
      System.out.println(s);
  }
}
```

break y continue

Dentro del cuerpo de cualquier estructura de iteración podemos controlar el flujo del bucle usando break y continue. break sale del bucle sin ejecutar el resto de la sentencias contenidas en él. continue para la ejecución de la iteración actual y vuelve atrás al comienzo del bucle para empezar una nueva iteración. Ejemplo:

```
for(int i=0;i<100;i++){

      // fuera del bucle

if(i==69) break;

            // próxima itera-
ción

if(i%9 != 0) continue;

System.out.println(i);

}

int i = 0;

// ¡Un bucle infinito !

while(true){

      i++;

      int j = i*27;

      // punto de sali-
da del bucle

if(j==1269) break;

      if(i%10 !=0)continue;

      System.out.println(i);

}
```

El odiado goto

La palabra clave goto ha estado presente en los distintos lenguajes de programación desde su aparición, goto nacía de la forma de control de flujo en la programación en ensamblador: si condición A, entonces salta a una etiqueta, de otra manera salta a otra etiqueta. Sin embargo goto es un salto incondicional,

Goto nacía de la forma de control de flujo en la programación en ensamblador: si condición A, entonces salta a una etiqueta, de otra manera salta a otra etiqueta

es decir, siempre se realiza el salto; luego, surge la duda de si en todas las ocasiones se produce el salto. ¿No habrá siempre alguna manera de organizar el código para no incluir estos saltos, que tienden a hacer la lectura del programa una odisea? El matemático Edsger Dijkstra publicó "Goto considered harmful" y desde entonces goto es una especie en vías de extinción. El único uso útil del goto y para lo cual es utilizado en C es para escapar de algún

LISTADO 1

```
//: BitOperators.java
//: Demostración de los bitwise operators de Java
import java.util.*;
// Clase para producir retardos
class Delay{
        public void start(int seconds){
                long t = System.currentTimeMillis()
                                + seconds*1000;
                while(System.currentTimeMillis() " t)
                        ;// no hacemos nada
        }
}
public class BitOperators
{
  public static void main(String args[])
  {
    // Generación de un par de n aleatorios.
    Random rand = new Random();
    int i = rand.nextInt();
    int j = rand.nextInt();
    // Utilización de los operadores.
    DecToBin("-1", -1);
    DecToBin("+1", +1);
    // Valores máximos y minimos del tipo int
    DecToBin("maxpos", Integer.MAX_VALUE);
    DecToBin("maxneg", Integer.MIN_VALUE);
    DecToBin("i", i);
    DecToBin("~i", ~i);
    DecToBin("-i", -i);
    DecToBin("j", j);
    DecToBin("i & j", i & j);
    // Para mejorar la visualización
    new Delay().start(10);
    DecToBin("i  j", i  j);
    DecToBin("i ^ j", i ^ j);
    DecToBin("i "" 5", i "" 5);
    DecToBin("i "" 5", i "" 5);
    DecToBin("(~i) "" 5", (~i) "" 5);
    DecToBin("i """ 5", i """ 5);
    DecToBin("(~i) """ 5", (~i) """ 5);
  }
  static void DecToBin(String s, int num) {
    System.out.println(
      s + ", Decimal: " + num + ", Binario: "
    );
    // Rutina de visualización de
    // números en formato binario.
    System.out.print("   ");
    for(int j = 31; j "=0; j--)
      if(((1 "" j) &  num) != 0)
            System.out.print("1");
      else
            System.out.print("0");
    System.out.println();
  }
}
```

Las construcciones con goto han sido erradicadas del lenguaje

fragmento de código en el que tenemos unos cuantos bucles anidados. En este caso si queremos salir de un bucle interior debemos usar unos cuantos break, que entorpecen la lectura del código.

Aunque goto es una palabra reservada en Java, no es usada en el lenguaje; Java no tiene goto. Pero tiene algo parecido, las palabras reservadas break y continue pueden llevar asociadas una etiqueta. Veamos un ejemplo:

```
Etiqueta:
// No podemos tener sentencias aquí
 Bucle-externo{
   Sentencias1
   Bucle-interno{
        Sentencias2
        break; //1
        // ...
        continue; //2
        //...
        continue Etiqueta; //3
        // ...
        break Etiqueta; //4
   }
  Sentencias3
 }
Sentencias4
```

En el caso //1 con break salimos del bucle interno ejecutándose Sentencias3 y volviendo de nuevo a comenzar el Bucle-externo. En el caso //2 con continue volvemos al principio del Bucle-interno ejecutándose Sentencias2. En el caso //3 continue Etiqueta salta fuera del Bucle-interno y del Bucle-externo, y comienza de nuevo la iteración en el Bucle-externo, ejecutándose Sentencias1. En el caso //4 break Etiqueta, salta también fuera del Bucle-interno y del Bucle-externo, pero en este caso no entra en el Bucle-externo, ejecutándose por tanto Sentencias4.

LISTADO 2

```
// DecToRom:   Conversión de números decimales
//             a números romanos.
// Uso: java DecToRom numero_decimal
public class DecToRom{
// Declaración de una constante estática
public final static int L = 100;
int v[] = { 1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};
String[] r = { "M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
String str = "";
// Contenedora del n romano
// Función de conversión de decimal a romano.
public boolean Convers(int number)
{
  for(int i=0;i"v.length;i++)
  {
            while(v[i] "= number)
            {
                        if((str.length() + r[i].length()) "L)
                        {
                                    System.out.println(
                                    "Numero demasiado grande");
                                    return false;
                        }
                        str    += r[i];
                        number -= v[i];
                        }
  }
return true;
}
// Para favorecer la encapsulación de los datos
public String getRoman()
{ return str; }
public static void main(String args[])
{
  DecToRom T = new DecToRom();
  if(args.length " 1){
            System.out.println(
            "Uso correcto java DecToRoman número" );
            return;
}
// Los args. son cadenas y yo quiero un número
int num = Math.abs(Integer.parseInt(args[0]));
// Realizo la conversión
if(T Convers(num))
  {
            System.out.println("Numero romano = " +  T.getRoman());
  }
}
}
```

Juntándolo todo

Ahora es el momento de utilizar todo el conocimiento vertido a lo largo de este artículo.

En el listado 1 (correspondiente a BitOperators.java del CD-Rom) tenemos un programa que ilustra la utilización de los distintos operadores a nivel de bit con que cuenta Java, enseñándonos de paso la representación binaria en complemento a dos de los números enteros.

En el listado 2 (correspondiente a DecToRom.java del CD-Rom) podéis ver como ejemplo un programa que calcula el número romano equivalente al decimal que es entregado al programa mediante la línea de comandos. Los únicos conceptos nuevos que aparecen en él son: la definición de una constante mediante la palabra reservada final y el uso de los argumentos de la línea de comandos, cuyo uso difiere un poco del lenguaje C, donde args[0] contendría el nombre del programa, aspecto que Java ha variado.

En este artículo hemos avanzado bastante en la escritura de programas en Java. Por un lado hemos visto el tratamiento de los arrays que tiene el lenguaje. Hemos convertido a los programas en entidades que pueden variar su flujo de ejecución mediante el uso de sentencias condicionales y bucles. Y al final lo hemos juntado todo en un par de programillas, que nos han ilustrado unos cuantos detalles interesantes.

Bibliografía

-Gamelan: http://www.gamelan.com

-Bruce Eckel: http://www.EckelObjects.com/Eckel

-TeamJava: http:// www.teamjava.com

Contactar con el autor

Para cualquier duda, comentario, sugerencia o crítica se anima al lector a ponerse en contacto:

E-Mail myoprieto@redestb.es

LISTADO 3

```
// dinArray.java
// Relleno aleatorio de un array y posterior ordenación
// mediante el método de selección.
import java.util.*;
public class dinArray{
static Random rand = new Random();
        // Generamos un entero entr 0 y (mod - 1)
static Integer pRand(int mod){
return new Integer(Math.abs(rand.nextInt()) % mod);
}
// Esta función no la podríamos haber
// realizado cont tipos int.
static void Swap(Integer a, Integer b){
Integer temp = a;
a = b;
b = temp;
}
// Algoritmo de ordenación
static void SelSort(Integer[] a)
{
int i=0, j=0, k=0;
Integer min = null;
for(i=0;i<(a.length-1);i++)
{
k = i;
min = a[k];
For(j=i+1;j<a.length;j++)
                // No existe la sobrecarga de operadores
                if(a[j].intValue()<min.intValue()){
k = j;
min = a[k];
}
a[k] = a[i];
a[i] = min;
}
}
// Función que rellena el array con enteros aleatorios
static void rellenoAleat(Integer[] ar, int modul){
For(int j=0;j<ar.length;j++)
ar[j] = pRand(modul);
}
// Función para imprimir los contenidos del array
static void prtArray(Integer[] ar){
        For(int j=0;j<ar.length;j++)
System.out.print(ar[j]+", ");
        // Si no enviamos un retorno de linea la
        // impresión en pantalla no se produce.
        System.out.println();
}
public static void main(String args[]){
Integer b[];
// Obtengo las dimensiones en tiempo de ejecución
b = new Integer[pRand(20).intValue()];
    rellenoAleat(b, 99); // Relleno el array
System.out.println("Desordenados: ");
prtArray(b);
    SelSort(b);        // Ordenación
System.out.println("Ordenados: ");
prtArray(b);
    }
}
```



Metodologías de programación en multimedia

Javier Sarsa Garrido

Los lenguajes de autor son lenguajes específicamente diseñados para crear y manipular contenidos multimedia. En este artículo nos adentramos en los mecanismos que posibilitan la escritura de aplicaciones multimedia, centrándonos en los distintos modos de programación que existen.

Algunos programadores os habréis planteado en alguna ocasión qué es eso de los lenguajes de autor. Esta disciplina no aparece todavía, salvo contadas excepciones, como parte del menú de asignaturas disponibles en las carreras técnicas. Pues bien, frente a la programación tradicional, basada en la escritura de listados en Pascal, C, ADA, Cobol o cualquier otro lenguaje, surgieron hace algunos años métodos de programación específicamente diseñados para crear y manipular contenidos multimedia. Muchos de estos métodos se apoyan en su propio lenguaje que, aunque similar en lo fundamental, difiere bastante de los clásicos en cuanto a sus posibilidades y orientación.

Como es lógico, los lenguajes tradicionales han sido la base sobre la que se han construido éstos de autor, por ello, aunque siempre vamos a poder crear multimedia con cualquier compilador, los de autor disponen de un conjunto de rutinas ya creadas que favorecerán el tiempo de desarrollo considerablemente.

En lo sucesivo analizaremos pues los mecanismos que nos permiten esta eficacia en el prototipado y escritura de nuestras aplicaciones multimedia; si bien lo haremos centrándonos en diferenciar los distintos modos de programación, sin entrar en las posibilidades creativas de los entornos.

¿Qué es exactamente un entorno de autor?

Un entorno de autor (al que nos referiremos por EDA) es un sistema preparado para favorecer el desarrollo de materiales multimedia interactivos. Hay que remarcar la palabra "interactivos" para diferenciar a éstos de programas como Microsoft PowerPoint, Astound o Adobe Persuasion, por ejemplo. Dichas aplicaciones en ocasiones son calificadas como multimedia, por cuanto permiten añadir texto, fotografías, sonidos, vídeos o animaciones, pero no permiten un potencial de interactividad alto, característica diferencial de los EDA. Por esto los programas antes mencionados están mal calificados y su categoría correcta sería la de programas de presentaciones. La interactividad, o comunicación bidireccional que se establece entre el usuario y la máquina, debe ser buscada siempre en un desarrollo multimedia. Los programas de presentaciones sólo permiten intercambios hombre-máquina primarios, como la pulsación de un botón, la navegación secuencial o directa, etc. Sin embargo, una interactividad compleja, como pudiera ser la resolución de un puzzle en la pantalla, es posible gracias a un buen lenguaje de autor.

Las facilidades otorgadas por éstos llegan hasta el punto que permiten pres-

cindir de conocimientos de programación y, en el peor de los casos, eliminan los problemas que supone la necesidad de dominar un determinado hardware, plataforma o sistema operativo en profundidad. Según algunos autores, el tiempo empleado en el desarrollo de un título multimedia es del orden de la octava parte empleando un EDA, del que llevaría hacerlo con un lenguaje clásico.

El tiempo empleado ayudándose de un sistema de autor, es del orden de la octava parte del que llevaría hacerlo con un lenguaje clásico

Las metodologías de programación

La reducción en el tiempo de desarrollo va intrínsecamente ligada al modo de programación que implante el EDA elegido. Establecer aquí una categorización exacta de los modelos de programación es más difícil que en los lenguajes clásicos, porque son las casas de software (en lugar de Universidades y organismos reconocidos) las que han marcado el avance en este sentido. Pero según diversos paradigmas de programación podemos hacer una clasificación de los EDA bastante adecuada a nuestros intereses:

Los EDA pueden estar basados en:

- Listados clásicos (scripts)
- Iconos o flujos icónicos (iconic flow)
- Escenas completas (frames)
- Actores, escenarios y listados (cast-score scripting)
- Tarjetas y listados (card scripting)
- Jerarquía de objetos (hierarchical objects)
- Enlaces hipermedia (hypermedia linkage)

- **Listados clásicos**

El listado tradicional, en multimedia script, es el paradigma utilizado por los entornos más potentes. Es el único caso que se puede calificar realmente como lenguaje de autor. Aquí se engloban algunos tan conocidos como Lingo (Director), Hypertalk (HyperCard), SuperTalk (SuperCard), OpenScript (ToolBook) o ScriptX (Kaleida).

Al estar la multimedia basada en objetos cada uno de ellos podrá albergar su propio código escrito. Esta separación de los distintos listados hace mucho más fácil la identificación de las rutinas que conciernen a cada elemento. El paradigma basado en listados tiende a ser más costoso en tiempo de desarrollo que otros (lleva dedicación escribir cada acción individual) pero, a su vez, también es el que nos da el más alto potencial de interactividad.

En los scripts pueden incluirse procedimientos, funciones, manejadores de eventos,... e intercambiar cualquier mensaje con otros elementos. Es similar, en la forma, a los listados clásicos con librerías de procedimientos que pueden existir en varios documentos separados pero que son susceptibles de añadir a nuestros proyectos. La diferencia es que ahora estos listados están ligados a objetos específicos del EDA, albergados dentro de los objetos propios del entorno elegido generalmente (campos de texto, botones, vídeos, pantalla completa, etc.). Estos lenguajes tienen la ventaja común de que su sintaxis no es tan complicada como la de C o siquiera la de Pascal. Se ha eliminado toda la abstracción arrastrada de lenguajes más antiguos, para sustituirla por la escritura en inglés llano. Por ejemplo, la misma pieza de código en lenguajes clásicos sería:

En lenguaje C:

```
int cuentaNumeros(Str255 cadena)
{
int i=0, cifras=0;
while (cadena[i]!=0)
{
  if ((cadena[i]>='0') & (cadena[i]<='9'))
  cifras++;
  i++;
}
return cifras;
}
```

En PASCAL:

```
function cuentaNumeros (cadena: str255): integer;
var i, cifras: integer;
begin
cifras:=0;
for i:=1 to integer(cadena[0]) do
  if (cadena[i]>='0') and (cadena[i]<='9') then
    cifras:=cifras + 1;
cuentaNumeros:= cifras;
end;
```

En LINGO:

```
on cuentaNumeros cadena
put 0 into cifras
repeat with i=1 to the number of chars of cadena
  put char i of cadena into ch
  if ch>="0" and ch<="9" then
    put cifras+1 into cifras
  end if
end repeat
return cifras
end
```

Como se observa, en el tercer caso han desaparecido la mayor parte de los elementos abstractos que existen en los dos primeros. Los puntos y coma que separan las instrucciones y las llaves ya no son necesarios. Son lenguajes similares al inglés hablado (aunque no recomiendo a nadie dirigirse así a una persona).

En definitiva el paradigma basado en listados, como bien sabemos los pro-

gramadores, es el más abierto y potencialmente nos proporciona mayores posibilidades. Pero puede quedar escaso si no va acompañado de un interface adecuado.

- **Iconos o flujos icónicos (iconic flow)**

Este sistema resulta bastante sencillo una vez habituados al aspecto de los iconos. En cuanto a su velocidad en tiempo de desarrollo, también es de los más rápidos.

El núcleo de este método lo constituye la paleta de iconos, en la que están alojadas todas las funciones e interacciones del sistema. Por otro lado, resulta también fundamental la existencia de una línea de flujo sobre la que se definen las relaciones entre los elementos y su lugar en el tiempo. Se trata pues de un modo de programación completamente visual, que resulta sorprendentemente potente cuando es combinado con la posibilidad de añadir scripts. Más adelante se verá un ejemplo de aplicación de esta técnica en el apartado dedicado a Authorware.

- **Escenas completas (frames)**

Este paradigma de desarrollo también incorpora una paleta de iconos, pero en este caso los enlaces establecidos entre los iconos son conceptuales y no siempre representan el flujo del programa.

Éste es también uno de los modos más rápidos de programación multimedia siempre que el programa disponga de funciones avanzadas de depuración.

- **Actores, escenarios y listados (cast-score scripting)**

Es el modelo más famoso de todos gracias a la consolidación lograda por el primer programa que lo implementó. Se basa en la metáfora que se puede establecer entre el desarrollo multimedia y un guión cinematográfico. Nuestro título se construye sobre una malla de casillas que representa un negativo de película. En esta malla las columnas son las escenas y las filas son los actores que participan en cada escena. Otra ventana agrupa a todos los actores (reparto) y pueden ser llevados al escenario trasladándolos sobre una casilla de la malla o sobre el espacio de la escena directamente. Se puede tomar como actor a cualquier elemento natural del proyecto: imágenes, textos, vídeos, animaciones, sonidos, etc.

Esta metáfora tan acertada ha sido muy bien aceptada por los desarrolladores, aun cuando los tiempos requeridos en generar una pantalla no son los óptimos.

- **Tarjetas y listados (card scripting)**

Este método es también muy conocido y uno de los más antiguos. La construcción se hace añadiendo directamente elementos sobre la pantalla, a la que se llama carta o tarjeta. A las tarjetas que coexisten como una sucesión indexada en el fichero se les puede hacer referencia mediante un identificador o un número de índice. Para entenderlo mejor, esta técnica también se basa en una metáfora, la del archivador.

En este caso las tarjetas son pantallas completas que pueden extraerse para ser consultadas. La mayoría de los programas que utilizan el método combinan además la posibilidad de añadir scripts a los elementos, dotando al EDA de amplísimas posibilidades.

- **Jerarquía de objetos (hierarchical objects)**

Está basado en una estructura OO y visualmente representada por objetos anidados e iconos de sus propiedades. Aunque la curva de aprendizaje no es sencilla, este método hace posible la construcción de materiales muy complejos.

La reutilización de éstos se facilita gracias a las propiedades hereditarias de los objetos.

- **Enlaces hipermedia (hypermedia linkage)**

Es similar al paradigma basado en escenas completas, en el que se muestran los enlaces conceptuales entre elementos. Sin embargo carece de la ventaja de que estos enlaces se realicen mediante una metáfora visual.

- **Un amplio abanico de opciones**

Elegir uno u otro programa de autor para nuestros fines puede llegar a ser una pesadilla. El movimiento que se ha generado en las compañías de software desde la llegada de los CD-ROM ha sido increíble y todas ellas pretenden consolidar sus productos (como es natural), confundiendo al usuario en un mar de promesas, algunas de ellas incumplidas.

Para que tengáis una idea de lo que existe ahí afuera, son más de 60 los entornos de autor que se pueden encontrar. De ellos aproximadamente la mitad corren sobre Windows, unos 25 sobre Macintosh, alrededor de 3 en UNIX y el resto sobre otros sistemas operativos.

En esta cuenta algunos entornos son comunes a varias plataformas y son ellos los más aconsejables, dada la gran utilidad de la "pluriplataformidad" en desarrollos multimedia dirigidos al mercado.

En la tabla 1 se han listado, además de los sistemas más conocidos, aquellos entornos que pueden encuadrar claramente en uno de los paradigmas de programación citados.

Como forma de comprender mejor a qué se refieren las distintas técnicas empleadas, vamos ahora a comentar los programas más representativos de cada una de ellas.

Director (actores, escenario y listados)

Observando la tabla 1 con detenimiento, podemos notar que sólo existe un

El número de sistemas de autor disponibles en el mercado es superior a 60

entorno que utilice el paradigma D (actores, escenario y listados). Se trata de Director. Este programa presenta una interfaz gráfica dividida en diversas ventanas. En la figura 1 podemos ver abierta la ventana de cast o reparto de los actores que pueden formar parte en la película de multimedia. Arrastrando cualquiera de los actores desde aquí hasta una casilla de la ventana score (escenario) haremos participar a dicho actor.

La flecha de la figura trasladaría el actor número 29 del reparto sobre la fila 20 de la columna 1 del escenario. Esto significa que automáticamente este actor forma parte de la escena 1, junto al resto de los actores. En Director es posible que intervengan hasta un máximo de 48 actores por escena, es decir, existen 48 filas en la ventana score. Los elementos a incorporar en nuestro escenario podrían ser cualesquiera de los que intervienen en multimedia; en este ejemplo la ventana cast muestra varias fotografías, un dibujo, textos y scripts.

La sucesión de las escenas producirá los cambios en la pantalla, de modo

TABLA 1. Entornos de autor y paradigmas de programación empleados. La P siginifica que el EDA dispone de un player para dicha plataforma. La F que estará disponible en un futuro.

Entorno	Casa de software	DOS	WIN	MAC	Otros	Paradigma
Apple Media Tool	Apple Computer		P	x		C
Authorware	Macromedia		x	x	NT	B
Course Builder	Discovery Systems Int.		P	x		B
CBT Express	AimTech		x			C
Click & Create	Corel		x		NT	C
cT	WorldWired		x	x	Unix	A
Digital Box Office	Power ProductionSoft.			x		C
Digital Chisel	Pierian Spring			x		E
Director	Macromedia		x	x		D
Everest Author	Intersystem Concepts		x			G
Expo	Paul Mace Software		x			A
Formula Graphics	Harrow Software		x		NT	G
GLpro	G-media	x	x			A
HM Card	University of Graz		x			G
HyperCard	Apple		F	x		E
HyperGASP	Caliban Mindwear			x		E
HyperSense	Thoughtful Software				Next	E
HyperStudio	Roger Wagner Pub.		x	x		E
HyperWriter	Ntergaid	x	x			C
IconAuthor	AimTech		x	P	NT,OS/2,Unix	B
MaestroP ro	OmniMedia plc		x			G
MetaCard	MetaCard		x		NT,Unix	E
Microcosm	Multicosm,Ltd.	x	x			G

(Continúa)

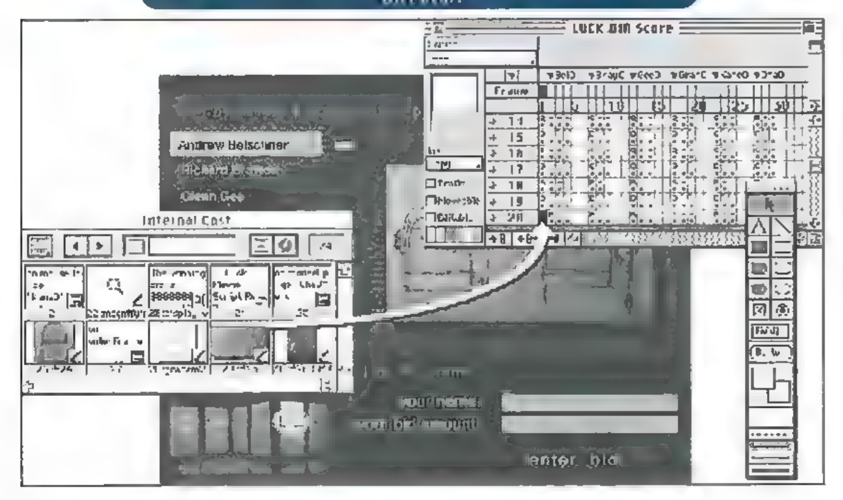
TABLA 1.

Entorno	Casa de software	DOS	WIN	MAC	Otros	Paradigma
MMD	Capella Computers Ltd		x			G
MocKingbird CBT	Warren-Forthought,inc.		x	x	NT	A
mTropolis	mFactory		P,F	x		F
OpenInfo Manager	Horizons Technology		x			C
Oracle Media Objs.	Oracle		x	x		E
Podium	Delaware University		x			G
PowerMedia	RAD Technologies		x	x	NT,HP,Sun,SG	C
Quest5 for Win.	Allen Communications		x			C
ScriptX	Kaleida Labs		x	x		A
Story Space	Eastgate Systems		x	x		A
SuperCard	Allegiant			x		E
SuperLink	Alchemedia		x			E
TenCORE	Computer Teaching Corp.	x	x			A
Toolbook	Asymetrix Corp.		x			E

que la realización de variaciones de posición de los actores sería una forma de lograr la animación de éstos sin recurrir a la programación. Existe una línea temporal de avance hacia la derecha en el escenario que transcurre hasta que se coloca alguna orden de parada. Cuando una escena se detiene, el usuario puede actuar sobre ella; esto es lo que ocurriría por ejemplo en un menú de selección.

Además de utilizar esta metáfora en la construcción de los contenidos, Director tiene un lenguaje muy potente de programación. Denominado Lingo, permite crear listados (scripts) que, a diferencia de los lenguajes clásicos, van asociados a un objeto (o actor) en particular. Así, un dibujo, un campo de texto o cualquier otro elemento, lleva adherido su propio código.

FIGURA 1. El reparto (cast) y el escenario (score) en Director.

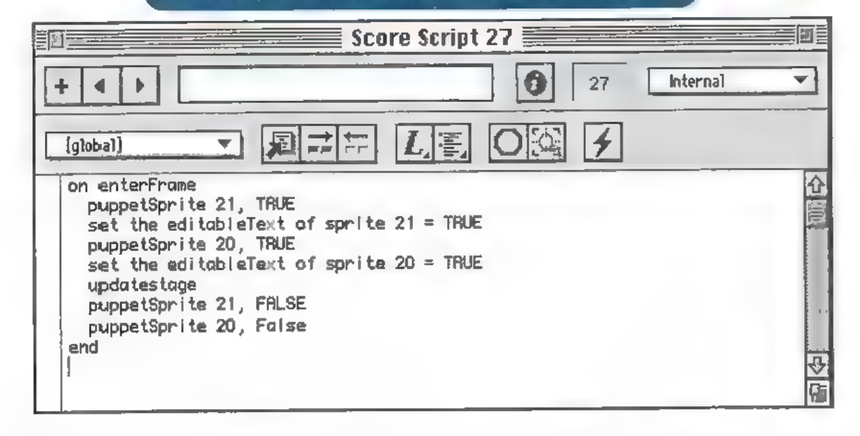


Pero un actor, como en el cine, puede aparecer en distintas escenas y comportarse en cada una de un modo diferente. Esto significa que podría obedecer a distinto código dependiendo de la escena en la que se encuentre. Al actor en una determinada escena se le denomina sprite y por tanto un sprite corresponde a una de las celdas del escenario. El código puede asignarse a estos tipos de objeto: al sprite (situación particular del actor); al actor, como ya habíamos mencionado; a la escena completa, cuyo código podrán llamar los actores en dicha escena y por último a la película completa (código visible desde cualquier otro).

Entonces, cuando el usuario realiza una acción, por ejemplo pulsar el ratón, si no existe código que gestione la pulsación en el sprite, el mensaje se envía al actor (cast), después a la escena (frame) y finalmente a la película (movie). Si todavía no se ha encontrado alguna respuesta, el mensaje se pierde.

Por ejemplo, al entrar en una escena se envía a su listado un mensaje "enterframe", (ver figura 2) que gestionará

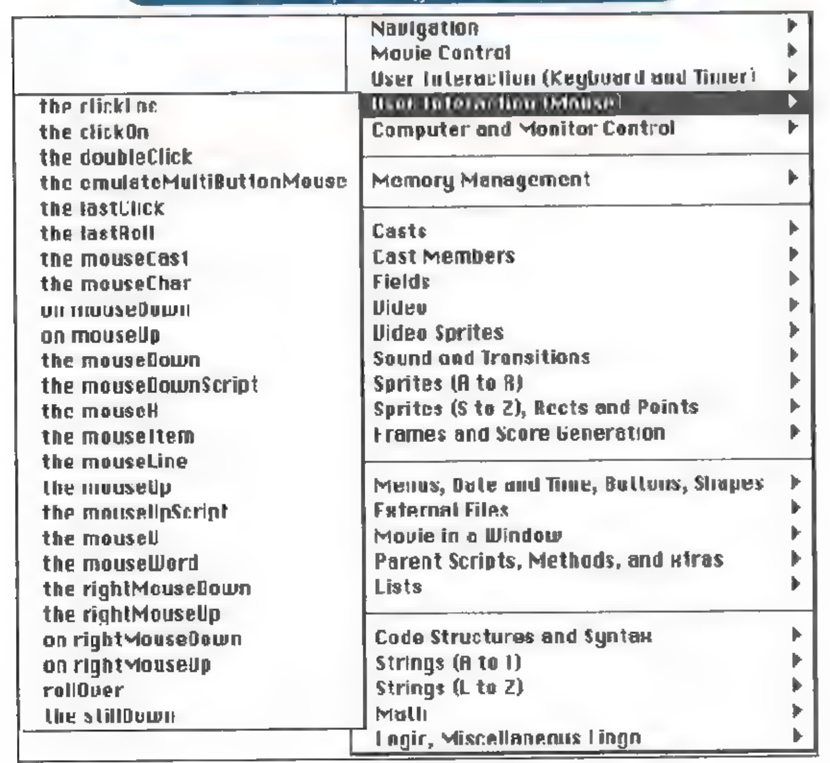
FIGURA 2. Ventana del listado de una escena



el correspondiente manejador "on enterframe". El lenguaje Lingo responde a multitud de eventos y aquí reside uno de sus potenciales. Podemos recoger las acciones de la pulsación-suelta del ratón o teclas, apertura-cierre-activación- movimiento de ventanas, entrada-salida de escenas, comienzo-parada de la película, control del tiempo, etc. Además existen funciones y propiedades de infinidad de objetos como aparece en la figura 3. En el caso de que a pesar de todo no pudiésemos satisfacer nuestras necesidades con las instrucciones incluidas en Lingo, Director todavía incluye la posibilidad de añadir código externo compilado para cada plataforma. Estos códigos reciben el nombre de Xtra, Xobj, Xcmd o Xfcn y pueden ser escritos en Pascal o C.

Este poder del Lingo y la facilidad proporcionada por la metáfora cinematográfica constituyen la base paradigmática de Macromedia Director.

FIGURA 3. Funciones y propiedades del Lingo por categorías.



Authorware (flujo icónico)

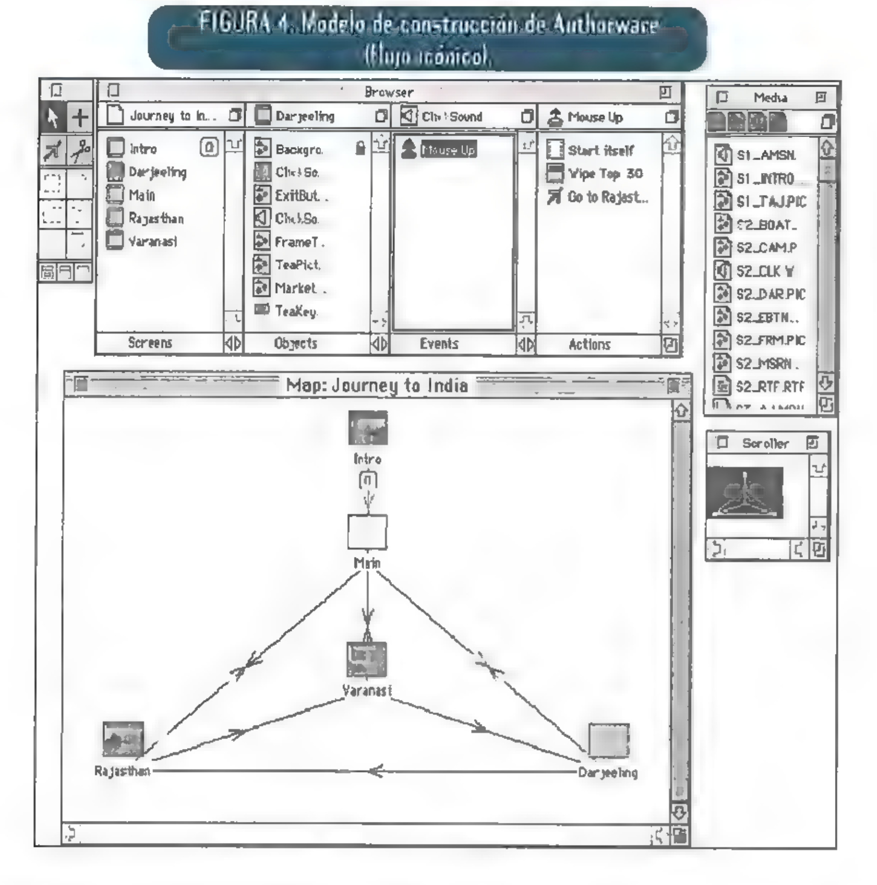
Aunque inicialmente era una herramienta de Asymetrix, más tarde fue adquirida por Macromedia con el fin de dominar el mercado multimedia como otra de las aplicaciones más potentes. Authorware basa su filosofía en la técnica de construcción basada en iconos.

Este programa presenta una ventana distinta por escena. Cada una de estas ventanas contiene su propia línea de flujo, sobre la que se van colocando los iconos necesarios, arrastrándolos desde la paleta (ver figura 4). Los iconos están preconstruidos para realizar una función determinada. Por ejemplo, siguiendo el orden vertical de la paleta el más alto de ellos sirve para importar una imagen; los sucesivos permiten el movimiento de un elemento, borrar la pantalla, esperar, navegar, crear estructuras de navegación, árboles de decisión, interacciones con el usuario, listados, mapas gráficos, incorporar vídeos digitales, sonidos y vídeo analógico. Una vez colocado el icono adecuado en un lugar preciso de la línea de flujo, un doble clic nos abre otra escena distinta o un diálogo de características del icono, en donde podemos configurar sus opciones.

El icono que contiene en su interior el signo '=' (igual) nos permite la incorporación de listados en un determinado lugar. Aquí no existe más que este contenedor para añadir scripts, a diferencia del método anterior en el que cualquier elemento podía albergar uno. En la figura 5 aparece un diálogo con una lista clasificada de las instrucciones del lenguaje de Authorware.

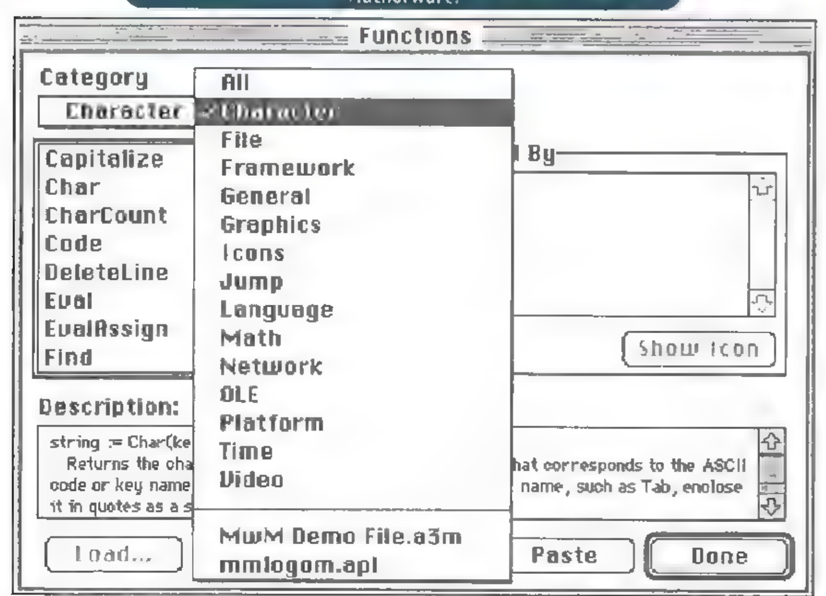
Destaca por tanto en este tipo de técnica la rapidez alcanzable en el prototipado gracias a un entorno altamente visual. Son muchas las aplicaciones que están basando de esta forma su filosofía, ofreciendo al usuario la carencia de un

FIGURA 4. Modelo de construcción de Authorware (flujo icónico).



lenguaje, como el aliciente de no tener que programar. El hecho de poseer funciones en iconos preconstruidos podría ser insuficiente o incómodo para algunos de nosotros, habituados a controlar todo el sistema. Sin embargo, el gran número de instrucciones de su lenguaje y la posibilidad (al igual que Director) de utilizar código externo compilado (Xtra, DLL, Xcmd, Xfcn) hacen de ésta otra herramienta de gran potencia.

FIGURA 5. Diálogo de selección de funciones en Authorware.



HyperCard (tarjetas y listados)

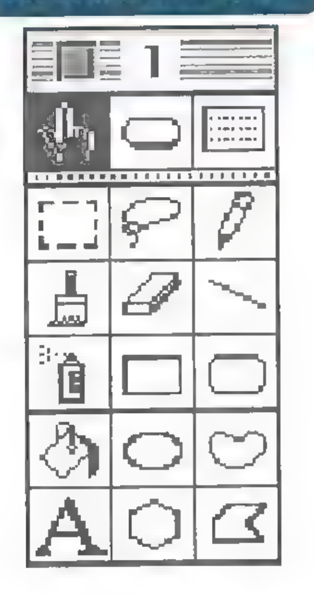
Se puede considerar como el precursor de la multimedia moderna. HyperCard nació en 1988 y desde entonces ha servido como base y orientación para muchos de los lenguajes de autor. Aunque hoy en día prácticamente ha desaparecido entre otros entornos más actualizados, la casa que lo fabrica asegura que este mismo año resurgirá de sus cenizas rejuvenecido.

Esta herramienta fue la primera en crear una estructura basada en medios (media). Botones, campos de texto e imágenes pueden ser combinados en la pantalla. La tarjeta, o la carta, es el nombre que se dio a cada una de las escenas, empleando la metáfora del archivador como paradigma de desarrollo. Así pues, hay que imaginar cualquier proyecto (llamados pilas) como un fichero de tarjetas en el que cada tarjeta es una pantalla y todas ellas están colocadas sucesivamente. De este modo podemos acceder a cada carta por su número de índice en el archivador.

HyperCard, nacido en 1988, es el precursor absoluto de los entornos de autor modernos

Una paleta de iconos nos permite escoger los elementos que van a ser incorporados en una tarjeta (figura 6). Cada uno de estos objetos puede ser situado en cualquiera de las dos capas existentes. La capa superior (card) varía con cada escena, mientras que la inferior (background) puede ser compartida por varias escenas

FIGURA 6. La paleta de HyperCard.



a la vez, con el consiguiente ahorro de tiempo en la colocación de elementos comunes.

Desde luego, lo que hace más potente a HyperCard es su lenguaje de programación, HyperTalk. Y su mecanismo de eventos que ha sido plagiado por varios entornos. Antes de explicarlo hay que saber que existen 6 niveles en los que colocar nuestro listado. Cuando un botón es pulsado recibe un evento mouseup, que de no recogerse es enviado a las sucesivas capas de objetos. De este modo llegará a la tarjeta, después al fondo, después a la pila, más tarde a la pila base (home) y por último al propio HyperCard. Si todavía no es interceptado se pierde.

Todas estas capas pueden tener su listado correspondiente y los procedimientos escritos en una capa son visibles sólo desde otros situados en el mismo nivel o desde niveles superiores. Esta manera facilita la reutilización de una sola rutina por varios botones, campos o cartas si la hemos escrito, por ejemplo, en el fondo.

mTropolis (jerarquía de objetos)

Es de los pocos entornos que utilizan de una manera seria la jerarquización de los objetos. La curva de aprendizaje de esta aplicación no es sencilla y puede llevar tiempo llegar a dominar bien su modo de programación. La aplicación incluye varias paletas de iconos preconstruidos (en este aspecto, similar al paradigma de iconos) y un lenguaje básico llamado Miniscript. Pero la diferencia fundamental del entorno estriba en la herencia, como en los lenguajes orientados a objetos.

En la figura 7 aparece la pantalla de un desarrollo de mTropolis. Los objetos (esta vez assets) son colocados directamente sobre la escena arrastrándolos desde un repositorio similar a la ventana de reparto de Director. A la izquierda otra ventana muestra la estructura jerarquizada del desarrollo completo, dividido en proyecto, secciones, subsecciones, objetos y propiedades de los objetos.

La definición de las propiedades y comportamiento de los objetos se hace escogiendo cualquiera de los 36 iconos destinados a ello y soltándolo sobre el objeto en cuestión (véanse las dos paletas de la derecha). Por ejemplo, para definir el estilo de un texto llevaríamos el icono que contiene una A sobre el texto. Después haciendo doble clic sobre el icono definiríamos un estilo completo para el texto y un nombre. De este modo podremos emplear dicho estilo sin necesidad de volver a hacerlo para cada texto. Entre estos 36 iconos existe la posibilidad de hacer transiciones, control de sonidos, modos de tinta, animación lineal, animación variable, medición del tiempo, etc.

Otro icono especialmente potente es el que define el comportamiento. Su aspecto es el de una máscara teatral. Llevando este útil sobre el objeto y pulsando en él se abre una ventana. En ella se crean las relaciones de comportamiento del objeto.

Como aparece en la figura 8, en el caso de un botón podemos definir su funcionamiento más complejo atendiendo a lo que hace el usuario. Por ejemplo, si pulsa el botón y suelta el ratón fuera de éste, la acción es inválida y el botón no ejecuta nada; al entrar o salir del botón con el ratón, éste se ilumina o apaga. Todas estas acciones quedan aquí defini-

FIGURA 7. Interfaz gráfica basada en objetos jerárquicos de mTropolis.

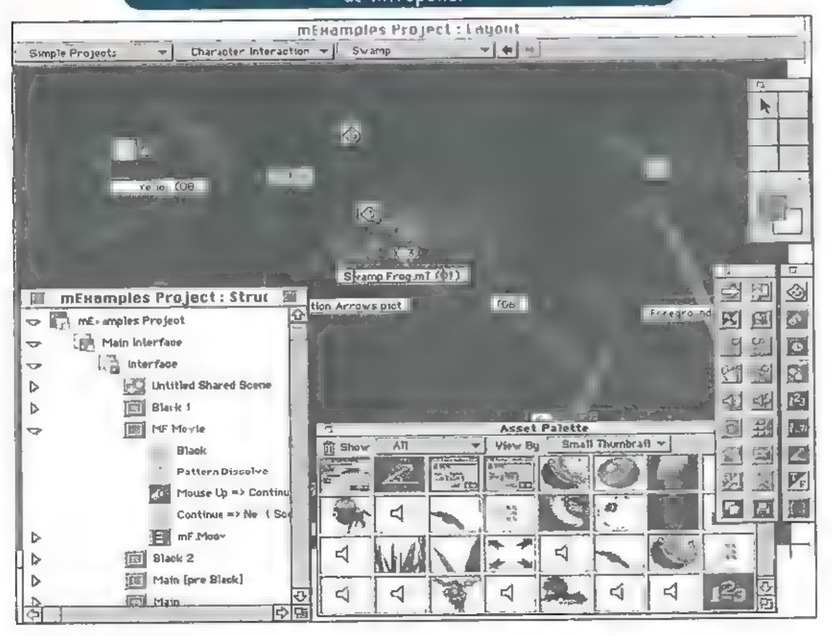
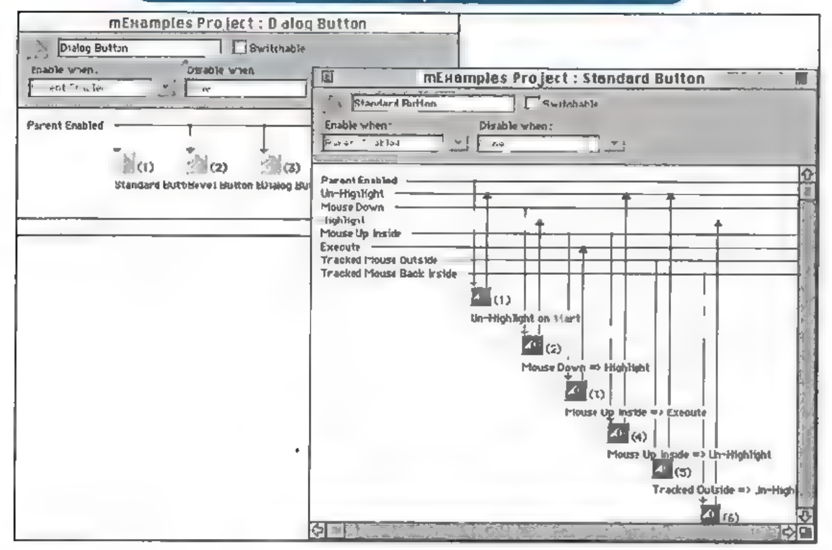


FIGURA 8. La definición de comportamientos y la herencia en mTropolis.



das y pueden ser reutilizadas para cualquier otro objeto. Además cada comportamiento podría contener a otros, obedeciendo a lo que son las características de herencia y jerarquía.

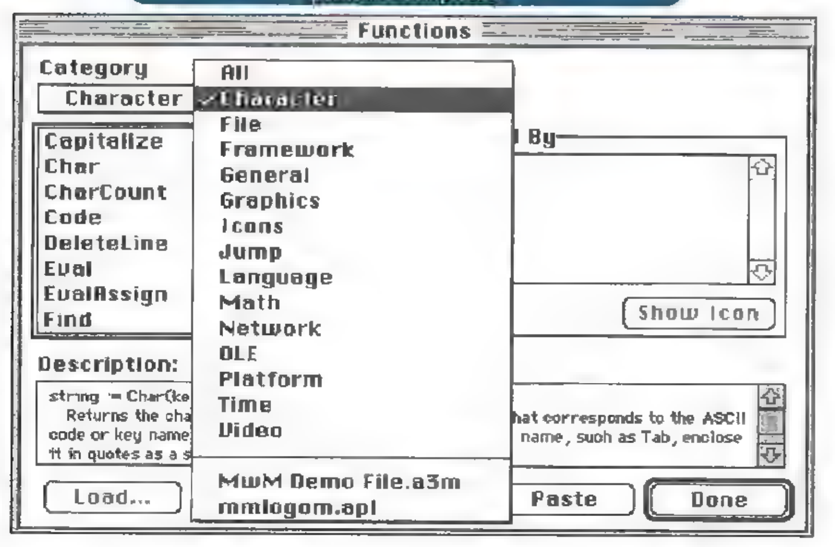
Apple Media Tool (escenas completas)

Como último caso veremos la implementación que hace AMT del paradigma centrado en escenas completas. Nuevamente existe un repositorio de objetos (media) del que se trasladan sobre la escena (figura 9). Otra ventana muestra la organización del proyecto dividiéndolo en escenas, objetos, eventos y acciones relacionadas con cada evento. Es posible definir todos los comportamientos en esta ventana.

Pero la diferencia fundamental con respecto al resto de los entornos es la de darnos una visión general del desarrollo con las relaciones existentes entre todas las pantallas. Es similar a los contactos que nos ofrecen los programas de presentaciones como PowerPoint, pero también nos muestra los flujos de navegación y los niveles de información del título multimedia.

En particular, para desarrollos muy extensos la telaraña que se genera es complicadísima y resulta poco útil y difícil de ver. AMT no dispone de un lenguaje propio, esta vez todas las acciones están preconstruidas y sólo podemos variar sus parámetros. Sin embargo otros entornos basados en escenas completas disponen de una interfaz gráfica distinta.

FIGURA 9. Apple Media Tool muestra las relaciones entre pantallas completas.



¿Qué entorno elegir?

La combinación entre una técnica visual y un lenguaje de autor extenso podría proporcionarnos las mayores prestaciones.

Como conclusión sería conveniente dar a los lectores un consejo sobre la elección correcta para crear contenidos multimedia. Sin embargo, esto siempre es arriesgado y depende mucho de la versión disponible para cada programa.

A pesar de todo, como regla general se puede decir que los entornos que se fundamentan en técnicas visuales permiten mayor rapidez de generación una vez que nos hemos habituado a ellos.

Por otra parte, los sistemas que incorporan lenguajes extensos suelen ser mucho más abiertos y versátiles que los anteriores.

Una combinación de ambos es la mejor opción. Quizás sea ésta la razón por la que Director y Authorware dominan el mercado.

Borland C++ Builder y la librería VCL



Arsenio Molinero Puente

En el número anterior de la revista, comentábamos las características del Borland C++ Builder, éste se basa en la programación visual, es similar al Visual Basic y al Delphi. Este entorno de programación, a diferencia de sus competidores visuales, genera código en C++; tenemos una herramienta orientada a objetos que aprovecha todas las características de este lenguaje.

Los controles y las formas son objetos, las clases de estos objetos se llaman VCL (Librería de Componentes Visuales). Con el producto se acompaña el código fuente de estas librerías, el cual está escrito en Pascal, pues las librerías se han portado del Delphi. En los manuales encontramos una parte que hace referencia a la equivalencia entre el código Pascal y C++, si deseamos convertir este código a C++ nos resultará muy fácil porque la programación de Objetos de C++ y Pascal es idéntica, es más, los comandos son los mismos.

El Borland C++ Builder es un producto que además de compilar código en C++ también compila código en Ensamblador y en Pascal, por tanto no es necesario convertir el código de las librerías para recompilarlo. Al ser C++ se pueden heredar las características de estos objetos para crear nuevos tipos de objetos y, por tanto, reaprovechar código.

Los controles

Como se comentó en el artículo anterior, tenemos 107 controles distintos, los cuales están divididos en 11 grupos temáticos para facilitar su localización.

Los controles tienen propiedades y eventos. Las propiedades son las características del control, es decir, el color, el ancho, el alto, la posición en la forma, el título, el nombre del control, etc. Los eventos son los mensajes que el control recibe generados por ciertos eventos producidos por el usuario o por el sistema como, por ejemplo la pulsación del ratón encima del control, la utilización del teclado o la carga del objeto en la pantalla. Cada control tiene internamente su gestor de mensajes como toda aplicación Windows, pero éste está encapsulado y es totalmente transparente.

Profundizando un poco más podemos darnos cuenta que las propiedades y los eventos son miembros públicos del objeto y las formas son las ventanas de Windows. Las propiedades son normalmente las variables y los eventos son las funciones. En la documentación que aporta el producto hay un fichero de ayuda que se llama *"VCL Reference"* en el cual podemos encontrar información referente a estos objetos. También encontramos dos libros, el *"Visual Component Library Reference Volume 1"* y el volumen 2, que componen un total de 1.800 páginas de información.

La potencia del Borland C++ Builder se basa en la gran cantidad de controles que tiene y en la utilidad de éstos, sin ellos el Borland C++ Builder sería un compilador de C++ normal y corriente y no podría competir contra el Visual C++.

Podemos modificar las propiedades y los eventos desde una ventana que se llama *"Object Inspector"*, esta ventana tiene tres partes bien diferenciadas: la lista de objetos, las lengüetas que se utilizan para indicar si queremos configurar las propiedades o los eventos y, por último, la parte de los datos. Si queremos configurar las propiedades de un objeto lo primero que tenemos que hacer es seleccionar de la lista el objeto y luego pulsar en la lengüeta de propiedades para que éstas aparezcan y las podamos cambiar; para los eventos se actúa de la misma manera.

Descripción de los controles

A continuación vamos a describir la mayoría de los controles que tiene el Borland C++ Builder, no pretendemos profundizar en ellos, pues se podría escribir un libro sobre éstos, pero sí destacaremos sus peculiaridades.

Grupo Standard

Como su nombre indica es el grupo que tiene los controles más utilizados para la gestión de captura y visualización de datos, y la creación de menús de aplicación.

Para poner un menú a una forma lo que tenemos que hacer es pulsar en la lengüeta *Standard* de la barra de herramientas y pulsar en el botón de control *Main Menu* y colocarlo en cualquier lugar de la forma, pues este objeto se visualiza siempre como barra de menú y su posición se localiza encima del área de trabajo de la ventana cliente. Éste es un objeto del tipo *TMainMenu* y tiene un número de propiedades muy pequeño: tenemos la propiedad *AutoMerge* que se utiliza para indicar que el menú se va a añadir al de la ventana hija, si su valor es false indica que el menú es normal. Esta funcionalidad no se utiliza con las aplicaciones del tipo MDI (Interface de Múltiples Documentos), pues ya tienen esa propiedad por defecto. También tenemos la propiedad Items que se utiliza para indicar los elementos del menú, si pulsamos en los puntos suspensivos del recuadro de la propiedad aparece una ventana desde donde podemos diseñar el menú. Esta ventana es de fácil manejo, simplemente con pulsar en la ventana de *Object Inspector* podemos darle las características de cada opción del menú; si deseamos añadir componentes simplemente tenemos que pulsar la tecla *Insert* y si lo que queremos es eliminar un componente, pulsaremos la tecla Supr. Otra propiedad es la de *Name*, que indica el nombre del objeto.

Cuando crea funciones para la gestión de eventos, el Borland C++ Builder las llama con el nombre que le hemos indicado y el evento a gestionar, por ejemplo, si ponemos en *Name* el valor *PruebaDeControl* y hacemos un doble click sobre el control, se creará una función que se llamará *PruebaDeControlClick*. Para los menús no existe ningún tipo de evento, sólo tienen eventos los componentes del menú.

También tenemos el control para crear menús emergentes (*PopupMenu*), estos son idénticos a los menús normales, con la diferencia de que se visualizan cuando el usuario pulsa con el botón derecho del ratón. Las propiedades de los menús emergentes son iguales a las de los menús normales añadiéndole *Alignment* y *AutoPopup*. La primera sirve para indicar dónde se visualizará el menú con respecto al ratón y la segunda, si tiene el valor true indica que con la pulsación del botón derecho del ratón se visualizara el menú. A diferencia de los menús normales, el menú emergente tiene un evento llamado *OnPopup*, éste se puede utilizar para realizar alguna tarea antes de visualizar el menú, pues se activa al pulsar el botón derecho del ratón.

Para poner texto en la forma se utilizan los objetos del tipo *Tlabel*, que están representados con el icono de Label (etiqueta). Este objeto tiene las propiedades típicas de configuración del texto, el tipo de alineación, el color, la posición y tamaño, etc. De todas las propiedades que trae destacan tres:

- *Cursor*, que nos permite indicar el cursor que tendrá el ratón cuando se desplace por encima del texto.
- *Font*, que se utiliza para indicar el tipo de letra a utilizar
- *Visible*, que nos permite visualizar u ocultar el texto a nuestro antojo en tiempo de ejecución.

Los objetos del tipo *Label* tienen bastantes eventos pero normalmente no se utilizan con las etiquetas, estos eventos los explicaremos con controles que hagan una mayor utilización de los mismos.

Si necesitamos un medio de introducción o modificación de texto por parte del usuario, podemos utilizar objetos del tipo *TEdit*, que están representados por el icono de Edit. El Objeto *TEdit* es uno de los que más propiedades tiene, además de las que hemos indicado para el objeto *TLabel* tenemos las relacionadas con el control del contenido del recuadro y las del aspecto del mismo. Las más interesantes son: *BorderStyle*, que indica si el recuadro de texto va ha tener borde o no; *CharCase* se utiliza para forzar el texto que se introduce a Mayúsculas, minúsculas o normal; *Enabled*, esta propiedad se utiliza para activar o desactivar el control; *MaxLeng*, mediante esta propiedad indicamos el máximo número de caracteres que se pueden introducir; *OEMConvert* indica si el texto introducido está en formato *OEM* (formato propio del MSDOS) o en el de Windows; *PasswordChar* es para que los caracteres de texto aparezcan camuflados con un asterisco, esto se utiliza para la introducción de contraseñas; y por último tenemos el *PopupMenu*, que permite indicar el menú emergente que queremos asignar a ese recuadro.

En lo referente a eventos disponemos también de una lista muy larga, de

esta lista destacan por su mayor utilización los siguientes: *OnChange*, este evento se utiliza para saber cuándo ha cambiado el contenido del control; *OnEnter* y *OnExit* indican cuándo reciben el foco de la aplicación y cuándo lo pierden respectivamente; *OnKeyDown*, *OnKeyPress* y *OnKeyUp* están relacionados con la utilización del teclado e indican que está pulsada una tecla, se ha pulsado una tecla y se ha soltado la tecla, respectivamente; *OnMouseDown*, *OnMouseMove* y *OnMouseUp* indican si está pulsado el botón del ratón sobre el control, si se mueve el ratón por encima del control y si se ha dejado de pulsar el botón respectivamente, también tenemos todos los eventos relacionados con el *DragDrop* (arrastrar y soltar), que explicaremos con controles más adecuados.

El control *Memo* es idéntico al *Edit*, solamente se diferencian en que el *Edit* se utiliza para introducir pequeñas cantidades de texto y el *Memo* para introducir grandes cantidades. Este control es el correspondiente a la clase *Tmemo* y tiene las mismas propiedades y eventos que el *TEdit*, destaca la propiedad *WordWrap* por las características del objeto; si tenemos la propiedad a true se forzará un retorno de carro al final de cada línea para que las palabras no se visualicen cortadas.

El control *Button* es uno de los que más se utilizan, la mayoría de las propiedades de éste ya las hemos explicado. Hay un par que se utilizan para identificar el tipo de botón: *Cancel*, con el valor a true indica que es un botón del tipo Cancelar; *Default* es igual que el anterior pero indica que es un botón del tipo por defecto.

Para permitir al usuario seleccionar entre varias opciones tenemos a nuestra disposición los controles *CheckBox* y *RadioButton*, el primero se utiliza para indicar el estado activado/desactivado; un ejemplo de utilización de este control sería, por ejemplo, si necesitamos pedir el estado civil del usuario (casado o soltero). El *RadioButton* se utiliza cuando tenemos más de dos opciones y sólo se quiere seleccionar una de ellas, por ejemplo para indicar el idioma en una forma *Wizard* (Castellano, Catalán, Gallego, Vasco). Para estos controles cabe destacar tres propiedades: *AllowGrayed* pone el control como desactivado; *Checked*, el control está seleccionado y por último State contiene el estado del control que puede ser *Checked*, *UnChecked* o *Grayed*.

Cuando necesitamos visualizar una lista de valores utilizamos un objeto del tipo *TListBox* o *TComboBox*; la *ListBox* visualiza a la vez varias líneas de la lista, la *ComboBox* es una mezcla entre *ListBox* y *Edit*, su forma es de Edit y cuando nos interesa seleccionar un valor podemos pinchar en la flecha de despliegue para que se visualice una lista de valores. Las propiedades a destacar son: *Columns*, que indica el número de columnas que tendrá la lista, por defecto tiene el valor 1; *ExtendedSelect*, si está a true permite seleccionar un rango de elementos consecutivos; *Items* permite introducir valores a la lista. Al pulsar en los puntos suspensivos del recuadro de la propiedad aparece una ventana *Edit* para que introduzcamos los valores de la lista; *MultiSelect* permite la selección de múltiples valores y por último *Sorted*, si tiene el valor a true ordena la lista alfabéticamente. La *ComboBox* tiene la propiedad *Text* que permite introducir el texto que queremos que se visualice en el recuadro Edit. Hay un evento que sirve para tener constancia de cuando se tiene que redibujar la lista, este evento se llama *OnDrawItem*.

El control *ScrollBar* se suele utilizar para indicar valores o desplazamientos, este control lo podemos ver en todas las aplicaciones realizando tareas muy dispares. El *ScrollBar* es un control que permite una amplia configuración; además de las propiedades ya conocidas tenemos las que indican el tipo de desplazamiento. Las más destacables son: *Kind*, para indicar la orientación de la barra; *LargeChange* indica el número de desplazamientos que realiza en un salto grande cuando se pulsa AvPag o RePag, y también cuando pulsamos con el ratón en cualquiera de los lados de la barra donde no está el recuadro de desplazamiento; *Max*, *Min*, utilizados para indicar el número mayor y el menor que representa la barra; *Position* indica la posición del recuadro de desplazamiento; *SmallChange* indica el número de desplazamientos que se realizan cuando pulsamos en las flechas de la barra. En lo que se refiere a eventos, aparece uno nuevo que se llama *OnScroll* e indica que se está realizando un desplazamiento en la barra.

Grupo Win95

Este grupo tiene los nuevos controles que trae Windows 95, controles que cada vez nos resultan más familiares.

Tab Control, con este nombre aparece un control muy útil, se suele utiliza para las ventanas de configuración de los programas. Es un control que simula un conjunto de fichas las cuales tienen una lengüeta por ficha y podemos colocar cualquier control dentro de cada ficha; cuando el usuario pulsa en una lengüeta aparece la ficha correspondiente. Este control además de las propiedades ya vistas tiene la propiedad *Tabs*, que permite indicar las fichas que componen este control. Al pulsar en los puntos suspensivos del recuadro de la propiedad aparece una ventana de edición idéntica a la de la propiedad Items de otros controles. En la parte de eventos observamos el evento *OnChanging* que se produce cuando se pulsa en una lengüeta.

El objeto *PageControl* es similar al *TabControl*, pero en lugar de gestionar fichas gestiona páginas, las páginas tienen sus propios eventos y propiedades. Tanto *PageContro* como *TabControl* permiten poner las lengüetas en cualquier posición. En este control cabe destacar las propiedades ActivePage, que indica la página activa, y *Aling*, que indica dónde se colocan las lengüetas.

El control *TreeView* se utiliza para visualizar listas de unidades, directorios y ficheros en formato de árbol. Este control tiene una propiedad que se encarga de asignar las imágenes a los tipos de datos del árbol, se llama Images.

ListView, control que se utiliza para ver listas de ficheros en formato de iconos, es el típico que aparece cuando pulsamos en el icono de Mi PC.

El objeto *THeaderControl* permite dividir una barra en varias partes, se utiliza en las hojas de cálculo para indicar las filas y columnas. Este control tiene una propiedad que se llama *Sections* que se usa para indicar las partes de la barra. Al pulsar en los puntos suspensivos del recuadro de la propiedad, aparece una ventana que nos pide el texto, el tipo y ancho de cada sección. También tiene eventos para gestionar la utilización de cada sección, estos eventos son: *OnResize* se produce cuando se cambia el tamaño de la barra; OnSectionClick, como el nombre indica se genera el evento cuando se hace un click en una de las secciones; *OnSectionResize* es idéntico al *OnResize* pero sólo para algunas de las secciones; *OnSectionTrack* se produce cuando al pulsar con el botón izquierdo del ratón en un lateral de una sección se intenta cambiar el tamaño.

Disponemos de un control para poder visualizar texto con formato Rich, este control se llama *RichEdit*. Gracias a este control nos podemos olvidar de las tediosas operaciones que necesitábamos realizar para poder ver textos de este tipo. En las propiedades se puede observar una preocupación por la visualización, permitiendo un control total sobre el recuadro de visualización. Los controles son: *HideScrollBars*, si tenemos esta propiedad a true se ocultan las barras de desplazamiento si no son necesarias; *PlainText* permite visualizar el texto con los códigos del propio lenguaje; *ScrollBars* indica si el control tiene barras de desplazamiento y de qué tipo; *WantReturn* permite que el usuario introduzca retornos de carro en el texto; *WantTabs*, igual que la anterior pero para los tabuladores.

Toda aplicación debería tener una barra de información. Existe un control para este tipo de barra, el *StatusBar*. Este control tiene la propiedad *Panels* que se utiliza para añadir las partes de la barra; si pulsamos en los puntos suspensivos aparece una ventana idéntica a la de *HeaderControl*. La propiedad *SimplePanel* indica si la barra es de un sólo panel (true) o de varios, si es de un sólo panel se tiene que utilizar la propiedad *SimpleText* para poner el texto del panel. La propiedad *SizeGrid* se utiliza para indicar si se puede cambiar el tamaño de la barra en tiempo de ejecución.

Grupo Additional

En este grupo encontramos controles que son de bastante utilidad, en él también podemos encontrar objetos que nos facilitan la gestión de algunos controles.

Los objetos *BitBtn* y el *SpeedBotton* son controles del tipo *Button*, que permite poner un bitmap en la cara del mismo. El primero te permite poner un dibujo y texto, y el segundo sólo un dibujo, este último es el que se suele utilizar en las barras de herramientas. La propiedad *Glyph* se utiliza para indicar el bitmap a pegar en la superficie del botón. También disponemos de la propiedad *Kind* que es donde se indica si queremos poner un dibujo propio (custom) o uno de los que ya trae el Borland. La propiedad *Layout* indica dónde se coloca el texto que acompaña al bitmap y la propiedad *Spacin* se utiliza para indicar la separación entre el dibujo y el texto.

Otro de los controles de este grupo es el *EditMask*, del tipo *Edit* pero con propiedades que permiten aplicar una máscara al texto que se introduce, por ejemplo para introducir un número de teléfono. Este control tiene la propiedad *EditMask* que permite seleccionar una de las máscaras que hay por defecto o crear una propia.

Los controles *StringGrid* y *DrawGrid* son posiblemente los más complejos y a su vez los que más nos facilitan la tarea de programación. Estos controles sirven para crear una hoja de cálculo, con todas sus características de diseño y operativilidad. Es evidente que un control tan potente tiene que tener infinidad de propiedades y eventos, éstas están camufladas para que no compliquen la utilización de las propiedades más comunes; las que más destacan son; *ColCount* que permite indicar el número de columnas que tiene la hoja; *DefaultColWidth, DefaultRowHeight* son para indicar el ancho y el alto de las columnas y filas; *FixedColor, FixedCols, FixedRows* sirven para indicar el color de las cabeceras de las columnas y filas y también para indicar cuántas columnas y filas componen la cabecera; *GridLineWidth* indica el ancho en píxeles de las líneas de separación; *RowCount* es la encargada de indicar el número de filas que tiene la hoja; y por último tenemos la propiedad Options que es la encargada de camuflar todas las propiedades de comportamiento de la hoja. Si queremos activar o desactivar estas propiedades tenemos que hacer un doble click en el signo "+" que tenemos a la izquierda de Options. En lo referente a eventos disponemos de una lista bastante grande pero la mayoría los hemos tratado ya, sólo hay dos nuevos: *OnColumnMoved* y *OnRowMoved*, encargados de avisar cuando se cambia el contenido de una columna o de una fila respectivamente.

Uno de los controles más curiosos es el de *Shape* que se utiliza para dibujar una figura. Cambiando la propiedad *Shape* indicamos si es un rectángulo, un círculo, una elipse, etc.

Grupo DataAccess

Este grupo es el que se utiliza para acceder a bases de datos, en él se encuentran los controles para indicar de qué base de datos estamos extrayendo la información, cómo tenemos que abrirla y las tablas que la componen.

Grupo DataControl:

Es el complementario del anterior, en él se encuentran los controles necesarios para gestionar el acceso a la base de datos desde los campos de entrada a los controles de visualización. Estos controles son similares a los ya estudiados, y no existen casi diferencias por tanto omitimos la explicación de los mismos.

Grupo Win3.1

Contiene una serie de controles que se pueden utilizar para Windows 3.1.

TabSet y *NoteBook* se utilizan conjuntamente para crear el mismo efecto que tiene Excel donde los documentos están compuestos por un libro y sus hojas. Para Windows 95 y Windows NT se utiliza el control *TabControl*.

El control *OutLine* sirve para visualizar varias líneas con formato de árbol. Este control es el utilizado en el Explorer para visualizar la lista de directorios y unidades. Tiene las siguientes propiedades: *Lines*, que permite introducir los elementos que componen la lista; *OutLineStyle*, esta propiedad indica cómo se visualizará la lista (sólo texto, texto con imagen, texto con imagen en formato de árbol ,etc.); *PictureClose, PictureLeaf, PictureMinus, PictureOpen, PicturePlus*, estas propiedades permiten indicar el bitmap que se utilizará para cada uno de los casos. Disponemos de un evento que nos avisa cuando se intentan meter más datos de los que podemos, este evento se llama *OnCollapse*.

El control *Header* es el equivalente al *HeaderControl*, para aplicaciones para Win95 y WinNT 4.0 se tiene que utilizar este último.

Grupo Internet

Como su nombre indica este grupo contiene los controles relacionados con el mundo Internet. Los controles que podemos encontrar en este grupo son bastante peculiares, pues lo más importante de cada uno son los eventos lo cual no se corresponde con los otros controles.

El objeto *TFTP* nos permite realizar operaciones de envío, recepción, borrado y visualización de ficheros. Este control tiene muy pocas propiedades: *RemoteHost* indica la dirección del servidor al que nos queremos conectar; *RemotePort* se utiliza para indicarle el puerto que utiliza el Servidor; *UserId, Password* se utilizan para poner el usuario y el password. En lo que a eventos se refiere existen para gestionar directorios y también cuando se abortan operaciones y borrado de ficheros.

HTML es el nombre del control que nos permite leer páginas de texto en formato HTML. Este control tiene la propiedad *ViewSource* que permite visualizar el código fuente de la página.

El objeto *THTTP* nos permite gestionar el HTML, permitiendo un control total sobre la visualización del documento. Tendremos que crear nosotros mismos las páginas, este protocolo simplemente se trae el documento o lo envía.

El *NNTP* es el control utilizado para la gestión de un cliente de news. Los eventos que trae están dirigidos a la gestión de artículos.

Los controles *POP* y *SMTP* son los que se utilizan para el correo electrónico. Las propiedades y eventos de estos controles son similares a los demás controles.

TCP y *UDP* son los controles utilizados para la establecer la comunicación, según el tipo de conexión tenemos que seleccionar el uno o el otro. Los eventos de estos controles están relacionados con la conexión entre máquinas.

Grupo Dialogs

Este grupo contiene los controles necesarios para llamar a los diálogos de Windows y así ahorrarnos el trabajo de crearnos nuestros propios diálogos. Los Controles que tenemos disponibles son: *OpenDialog* que se utiliza para seleccionar un fichero que abrir; *SaveDialog* es similar al anterior pero se utiliza para guardar un fichero; *FontDialog* sirve para seleccionar una de las fuentes instaladas en el sistema; *ColorDialog*, con este control aparece el diálogo de selección de colores; *PrintDialog* y *PrinterSetupDialog* son los correspondientes a la impresión y configuración de la impresora; *FindDialog* y *ReplaceDialog* son los típicos diálogos de búsqueda y sustitución de caracteres.

Grupo System

Este grupo contiene los controles necesarios para gestionar timers, archivos, multimedia, OLE y DDE.

El control *Timer* nos permite tener un gestor de tiempo para sincronizar operaciones. Es muy fácil de configurar pues tiene muy pocas propiedades: *Enabled*, que indica si está activo o desactivado; *Interval*, que indica el intervalo de mensajes en milisegundos. En lo referente a eventos, sólo tiene uno, *OnTimer*, que se activa cuando pasa el tiempo indicado.

MediaPlayer, con este nombre se identifica a uno de los controles más camaleónicos, simplemente cambiando la propiedad *DeviceType*, se cambia el tipo de sistema multimedia que se utiliza.

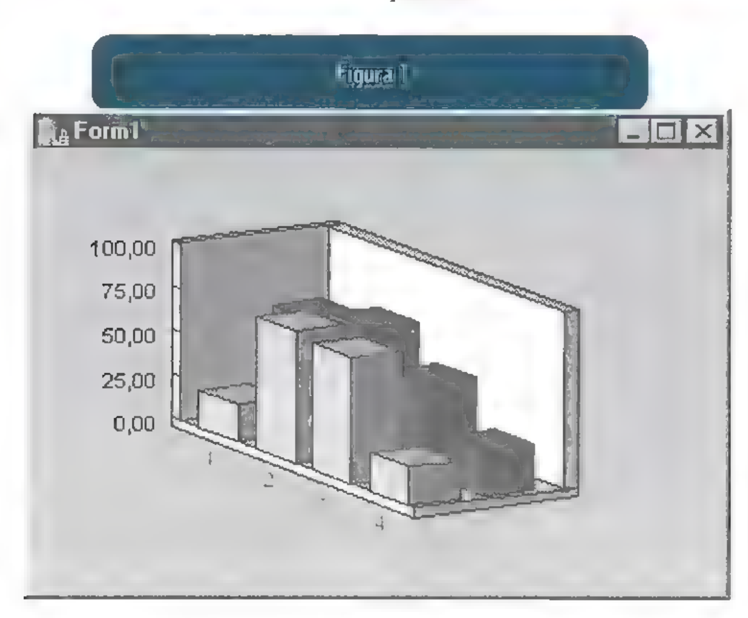
Grupo Qreport

Este grupo tiene los controles para realizar informes, es bastante útil si los informes a realizar tienen un formato fijo.

Grupo ActiveX

¿Quien no ha oído hablar a estas alturas del famoso ActiveX ?, pues aquí tenemos unos controles de este tipo de se manejan con mucha facilidad y son impactantes.

Figura 1



El control *ChatFX, GraphicsServer y VCFirstImpresion* nos permite crear gráficos de empresa con mucha facilidad, en la figura 1 podemos ver una forma en la cual se ha insertado un objeto del tipo *ChatFX*.

El objeto *TVCFormulaOne* es un control del tipo hoja de cálculo con todos sus componentes, tiene las celdas, las lengüetas y las barras de desplazamiento.

VCSpeller nos permite utilizar un diccionario para las correcciones ortográficas en nuestros programas.

Manos a la obra

Después de tanta teoría es hora de demostrar lo expuesto con una práctica, para ello vamos a hacer el típico programa que visualiza un reloj con la hora actual en pantalla.

Para empezar creamos un proyecto nuevo pulsando en el menú *File, New Application*. En la forma que tenemos por defecto hay que poner los controles necesarios para poder gestionar y visualizar la hora. En la figura 2 podemos ver la forma después de haber colocado los controles, para esta aplicación sólo hemos utilizado 4 controles, es un poco rústica pero para ver cómo se gestionan los eventos y las propiedades es suficiente. Los únicos controles que necesitan realizar operaciones son la forma y el timer, la primera por medio del evento *OnCreate* tiene que cargar la hora del sistema y actualizar los contadores.

```
//-----------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
struct time hora;
gettime(&hora);
contadorseg = hora.ti_sec;
contadormin = hora.ti_min;
contadorhora = hora.ti_hour;
}
//-----------------------------------------
```

El timer es el que comprueba los contadores y cuando uno se pasa lo pone a cero e incrementa el siguiente contador.

```
//-----------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
char temp[100];
if(++contadorseg > 60)
   {
   contadorseg = 0;
   if(++contadormin > 60)
      {
      contadormin = 0;
      if(++contadorhora > 23)
      contadorhora = 0;
      }
   }
sprintf(temp,"%02d:%02d:%02d",
      contadorhora,
      contadormin,
      contadorseg);
Label1->Caption = temp;
}
//-----------------------------------------
```

Para terminar la aplicación le colocamos dos objetos del tipo Tbevel para hacer un recuadro en 3D al reloj.

Tener un extenso conocimiento sobre los controles nos permite hacer aplicaciones de gran calidad con un esfuerzo mínimo. El Borland C++ Builder es un producto tan intuitivo que echándole un vistazo a las propiedades en los eventos de los objetos podemos manejar los controles como si los conociéramos de toda la vida.

Figura 2



Gestión de rendimiento en redes Novell



Enrique Díaz Trobo

En contra de lo que suele creerse, la elección del servidor y de la cantidad de memoria con que dotarle, no son las decisiones más críticas que ha de adoptar un administrador de red, ya que hoy en día es posible acceder a las últimas tecnologías a un precio razonable. El abaratamiento del precio de los equipos hace de ello un tema casi trivial. Un PC con procesador Pentium 200, con 16 ó 32 megas de RAM y 1 ó 2 Gigas de almacenamiento en disco haría un buen servidor departamental, y ésta es una configuración normal para un equipo de uso doméstico.

Si, por ejemplo, decidiéramos montar un servidor con 32 Megabytes de RAM y resulta que nos hemos quedado tan cortos que fuera necesario llegar a doblar esa cantidad, la actualización no supondría ningún problema, y su coste sería muy bajo, ya que el precio de las memorias no llega a 1.000 pts. el mega. Esto es así siempre y cuando nuestro servidor no vaya a soportar cientos de conexiones, en cuyo caso sería mejor que optáramos por un superservidor, mucho más caro, y por tanto deberíamos poner especial cuidado en su elección. Incluso en este caso, hay que valorar con mucho cuidado la posibilidad de instalar un superservidor o bien múltiples servidores de bajo coste.

Una buena planificación previa de la red nos aproximará bastante a los requerimientos necesarios, pero sólo su puesta en marcha y su uso efectivo nos dará la medida exacta de nuestras necesidades. Lo que sí nos permitirá la planificación es identificar los cuellos de botella que reducen el rendimiento general de la red y minimizar su impacto en el futuro. Los cuellos de botella típicos suelen ser una topología o cableado inadecuados, componentes del servidor como dispositivos de disco o tarjetas de red, e incluso ineficiencia de las estaciones de trabajo. Puede ocurrir incluso que el rendimiento caiga 'misteriosamente' a causa de un único usuario que ocupa casi todo el ancho de banda disponible en la transmisión de grandes archivos o bien realiza tareas de cálculo intensivo.

Algunos métodos para evitar estos cuellos de botella son:

Utilizar la tecnología Fast Ethernet, que permite alcanzar velocidades de transmisión de 100 Mb/seg.

Segmentar la red, si es preciso, añadiendo una o varias tarjetas de red adicionales de alto rendimiento en el servidor. Si hay usuarios que requieren un gran ancho de banda (por utilizar aplicaciones multimedia, por ejemplo) darles un segmento de red exclusivo e incluso asignarles un servidor de mayor rendimiento que al resto de usuarios de la red.

Utilizar discos duros SCSI. Ofrecen mayor rendimiento que los EIDE y la diferencia de precio no es razón para no utilizarlos.

Si en ocasiones es difícil decidir qué componente actualizar para mejorar las prestaciones de un PC doméstico, incrementar el rendimiento de una red puede ser una tarea ardua. En este artículo veremos cuáles son los componentes de la red a los que deberemos prestar una especial atención, así como el manejo que los servidores Novell hacen de la memoria.

- Asignar las aplicaciones que ocupen mucho el procesador a un servidor diferente al que acceden habitualmente la mayoría de los usuarios.

Configuración del servidor

Hemos visto cómo la elección del servidor no ha de plantearnos problemas, ya que el mercado está repleto de equipos baratos y con buenas prestaciones, pero sí hemos de tener en cuenta una serie de consideraciones para minimizar el impacto de los posibles cuellos de botella que se producen en el propio servidor. Estos cuellos de botella se producirán entre la memoria y las placas de red, CPU y disco duro.

Es importante tener en cuenta que la sobrecarga de un servidor no sólo se produce en la CPU. En un servidor de ficheros típico la carga se concentrará en las tarjetas de red y en los canales de entrada/salida del disco duro; en muchos casos estos dos apartados suponen un 90 por ciento de la carga del servidor. En los servidores de archivos el servidor simplemente pone a disposición de los usuarios los ficheros, con lo que la actividad del procesador no es muy grande.

Si nuestro servidor va a ser de aplicaciones cliente-servidor, un motor de base de datos por ejemplo, las mejoras las obtendremos principalmente por el lado del procesador.

Para decidir la configuración del servidor tendremos en cuenta si su actividad ocupará más los canales de entrada/salida o bien el procesador.

En el caso de que el servidor haga un uso intensivo de la CPU deberíamos:

- Instalar placas de red con control de bus, que pueden leer y escribir en el disco del servidor obviando el procesador.
- Comprar el procesador más rápido que nos permita el presupuesto o instalar un sistema multiprocesador.

En caso de que el servidor sea de archivos:

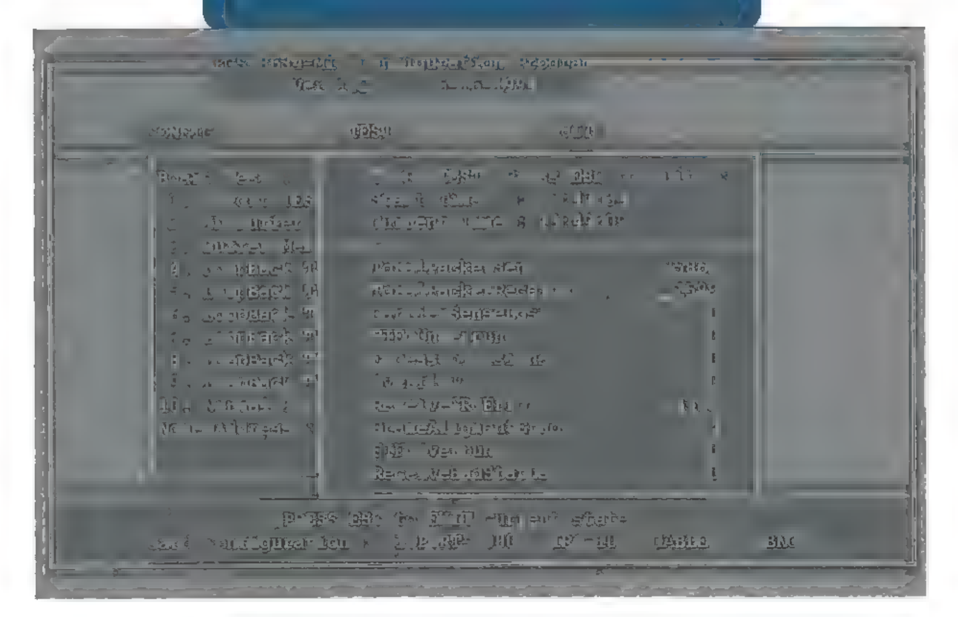
- Utilizar equipos con bus PCI.
- Utilizar tarjetas de disco SCSI, si es que no las teníamos ya.
- Incrementar la memoria caché y/o RAM.

Rendimiento global de red

Las redes locales no tienen un tendón de Aquiles. Tienen montones de ellos. A continuación detallo algunos que debemos repasar si nuestra red no tiene un rendimiento adecuado.

- **El servidor.** Si tenemos algún 386 ó 486 con bus que no sea PCI, no tenemos que darle muchas vueltas. Reconvertirlo en una estación de trabajo y comprar un Pentium con bus PCI.
- **Control de bus.** Consiste en relevar al procesador de ciertas tareas, haciendo que esas tareas sean llevadas a cabo por un procesador 'especialista' situado en otra tarjeta. Una tarjeta de red con control de bus escribe datos en memoria o disco sin ocupar al procesador del servidor en ello.
- **Memoria caché.** Reduce el tiempo de intercambio entre el procesador y la memoria. Entrada/salida de disco. Es de los puntos más importantes que considerar.
- Controladoras SCSI y discos duros rápidos son fundamentales para que un servidor rinda adecuadamente.
- **Las tarjetas de red.** La del servidor que sea PCI y de 32 bits; si es posible con técnicas avanzadas de buffering y control de bus. Para las

FIGURA 1. Una conexión en mal estado provocará errores en la transmisión de datos y disminuirá el rendimiento de red.



estaciones de trabajo, las más rápidas que nos permita el presupuesto y, si puede ser, en bus PCI.

- **El cableado.** Es lo primero que hay que mirar cuando el rendimiento de la red cae sin prisa pero sin pausa. Si hemos añadido estaciones de trabajo debemos ir pensando en cómo segmentar la red. Esta solución también es muy útil cuando hay unas pocas estaciones que generan gran cantidad de tráfico en la red, dándoles su propio segmento de red. Si estamos utilizando aplicaciones multimedia, cambiar el cableado a uno que soporte mayor tráfico, como Fast Ethernet. Si tenemos una red ya estable, de cierta edad, y en la que no hemos tocado nada, la caída del rendimiento puede deberse a que el cable y alguna tarjeta de red 'se están muriendo'. En el caso del cable, los tropezones, los movimientos al desplazar la caja del ordenador y los golpes de escoba del encargado de la limpieza, hacen que el cable se deteriore sin llegar a fallar del todo, de modo que la comunicación entre el servidor y algunas estaciones no es limpia, con lo que se reenvían continuamente paquetes de información que llegan en mal estado y se produce un incremento significativo en el tráfico de la red. Deberemos pasar un test de comunicaciones entre el servidor y todos sus nodos, renovando aquellas conexiones con un alto índice de errores en la transmisión. En el caso de las tarjetas de red, normalmente es debido a picos de tensión que deterioran los chips y los hacen fallar sutilmente. Para estos casos todas las tarjetas de red suelen incluir programas de autocomprobación. Dependiendo de la calidad de sus componentes, todo el ordenador debe estar más o menos en el mismo estado. Utilizar si es posible regletas que protejan contra sobretensiones.

Optimizar el servidor

Otra manera de mejorar el rendimiento de la red es modificar las asignaciones de memoria que los servidores Novell reservan para sus procesos de optimización. Para saber qué parámetros del servidor debemos variar nos serviremos de la utilidad monitor que proporciona Novell. Pero antes de verlo en detalle debemos saber cuáles son los procesos de optimización que utiliza Novell y cómo maneja la memoria.

Además de la conocida FAT, los servidores de Novell utilizan una tabla adicional llamada DET (Directory Entry Table, tabla de entradas de directorio), existe una DET para cada volumen y contiene las entradas de directorio para ese volumen. Una entrada de directorio consiste en información básica sobre los ficheros. En cada entrada se almacena la siguiente información:

- Fichero (nombre, propietario, fecha de última actualización y el primer bloque del disco duro en que se almacena el fichero).
- Derechos sobre el fichero.
- Derechos sobre el directorio.

Los procesos de optimización utilizados por Netware y que afectan al uso de la memoria son:

- *Directory Caching* (caché de directorio). Copia la FAT y parte de la DET en la memoria RAM del servidor.
- *Directory Hashing* (ordenación de la tabla de directorio). Ordena la DET que se ha cargado en memoria, de esta manera el sistema puede encontrar las direcciones correctas examinando sólo algunos directorios o entradas de fichero. Este proceso disminuye en un 30% el tiempo empleado por el disco en sus procesos de entrada/salida.
- *File Caching* (caché de archivos). Almacena en memoria los ficheros o programas más frecuentemente usados. El acceso a estos ficheros se realiza al menos 100 veces más rápido.

Para estos procesos de optimización se reservan áreas de memoria; *File Cache Buffers* (búferes para caché de archivos) para el *File Caching; Directory Cache Buffers* (búferes para el caché de directorio) para el *Directory Caching* y *Directory Hashing*; y *Packet Recive Buffers*, para almacenar paquetes de datos enviados por las estaciones y que no se pueden procesar en ese momento.

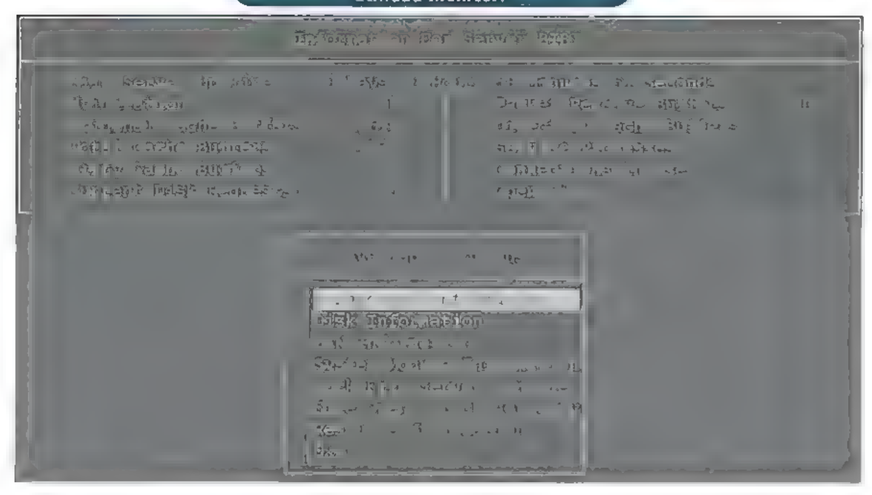
Veamos ahora cómo maneja Netware todo esto. La memoria se subdivide en Pools, que son porciones de memoria que se utilizan en función del uso que hará de ella.

La principal de estas porciones o pools es File Caché (cache de archivos).

Cuando el servidor arranca, toda la memoria es asignada a este área. Al ser el área principal, abastece a otras subdivisiones de manera que cuando se incrementa la demanda para otros recursos, el numero de búferes de caché para archivos disminuye. Las áreas abastecidas por File Cache son:

- Área de memoria permanente. Se usa para almacenar el *Directory Cache Buffers* (caché de directorios) y el *Packet Recive Buffers* (caché para paquetes recibidos) y otra información que tiene que estar cargada de forma permanente, como por ejemplo las asignaciones de disco. El sistema operativo le asigna la mínima cantidad posible de memoria, ya que éste no devuelve memoria al área principal pero le permite crecer dependiendo de las necesidades.
- Área de memoria semipermanente. Se usa para pequeñas cantidades de memoria requeridas por módulos que se prevé serán usados durante mucho tiempo, como por ejemplo los controladores de disco y tarjeta

FIGURA 2. La pantalla principal de la utilidad *monitor.*



de red. Este área devuelve la memoria no utilizada al área principal.

Dentro del área principal existen dos divisiones:

Caché movible. Se abastece directamente del área principal y se usa para almacenar tablas que crecen dinámicamente, como la DET y la FAT. Se llama 'movible' porque los bloques de memoria utilizados cambian de lugar para evitar la fragmentación dentro del área principal de memoria.

Caché no movible. Se usa para los módulos cargables (NLM's) y asignar grandes búferes de memoria.

Con esta información en la mano, estamos ya en disposición de comprender la información que nos ofrece la utilidad monitor, evaluar dicha información, aislar cuellos de botella y ajustar los parámetros del servidor para adecuarlo lo más posible a su situación particular.

La utilidad monitor

La utilidad monitor puede ofrecernos una gran cantidad de información. Aquí sólo veremos aquellas que pueden ser importantes para evaluar el rendimiento del servidor y el uso que se hace de la memoria.

TABLA 1. La tabla de velocidad de transferencia descarta cualquier duda respecto al bus a elegir para el servidor

Tipo de bus	Tamaño	Rendimiento
ISA	16 bit	1,5 Mb/seg
Microchannel	32 bit	20 Mb/seg
EISA	32 bit	33 Mb/seg
PCI	32 bit	132 Mb/seg

La pantalla principal de monitor nos muestra la siguiente información:

Utilization (utilización). Muestra el porcentaje de utilización de la CPU del servidor. Con el servidor recién arrancado, y cargando tan sólo controladores de disco y tarjeta, su valor debe ser 0. Este número va cambiando según se conectan y desconectan usuarios y según trabajen con aplicaciones o archivos. En general, no se debe permitir que este valor supere el 80%; en caso contrario puede que sea necesario migrar a un procesador más potente o trasladar aplicaciones y usuarios con mucha carga de trabajo a otro servidor.

Server Up Time (tiempo en marcha del servidor). Aunque parezca trivial, esta información es importante para evaluar correctamente la información que nos ofrece monitor. Netware va ajustando con el tiempo su configuración a la carga de trabajo de sus nodos y a sus propios procesos internos, de modo que la información mostrada por un servidor que lleva sólo algunas horas en marcha puede no ajustarse a los requerimientos medios del sistema.

Total Cache Buffers (Búferes totales para caché). Este número muestra el número de búferes que actualmente están disponibles para *File Caching* (subir a memoria los archivos más utilizados). Los búferes para caché son muy importantes para el rendimiento del servidor. Si se nos va muy abajo el rendimiento decrecerá rápidamente. Para obtener el porcentaje de memoria disponible para los búferes de caché consultaremos en *Resource Utilization* (utilización de recursos) la opción *Server Memory Statistics* (estadística de la memoria del servidor). Si este porcentaje se encuentra por debajo del 20%, es necesario añadir memoria inmediatamente. El valor ideal debe estar en torno al 50%.

Dirty Cache Buffers (búferes de caché modificados). Este valor indica el número de búferes de caché que contiene información que ha sido modificada y están esperando para ser escritos en el disco. Si este valor llega al 75% del valor de los búferes totales para caché, deberíamos incrementar el parámetro *Maximun Concurrent Disk Cache Writes* de la orden set. El valor de *Dirty Cache Buffers* es un valor, no un número, de modo que hemos de ser nosotros los que calculemos el porcentaje. Por ejemplo, si el valor de *Total Cache Buffers* es 900 y el de *Dirty Cache Buffers* es de 782, podemos

estimar un 75%, ya que 782 es un número que supera ampliamente las tres cuartas partes del total.

Packet Receive Buffers (Búferes para paquetes recibidos). Nos muestra el número disponible de búferes para dar soporte a las peticiones de las estaciones de trabajo. Es posible decrementar la necesidad de estos búferes incrementando el valor de *Service Processes.*

Directory Cache Buffers. (Búferes para el caché de directorios). Número de búferes dedicados al caché de directorios. Si sobrepasa los 100 es necesario incrementar el valor del parámetro *Minimun Directory Cache Buffers* de la orden set.

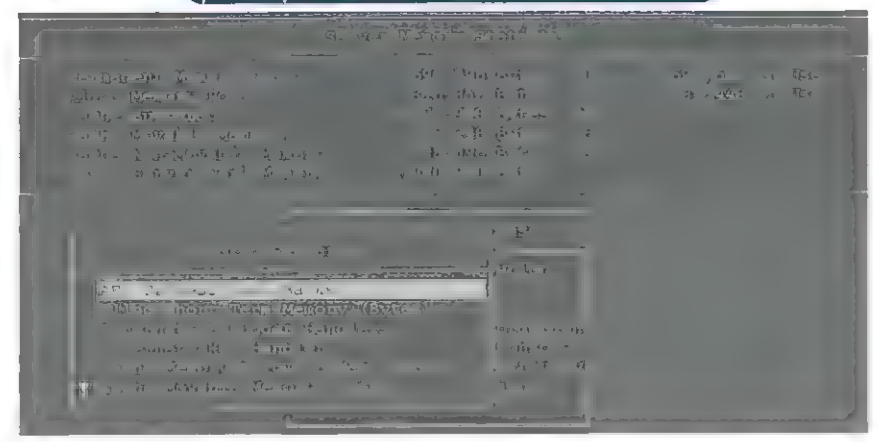
Service Processes. Representa el número de 'manejadores de tareas' asignado para las peticiones de las estaciones. Si este número se acerca a 20 y se tiene suficiente memoria, se debería incrementar el valor del parámetro *Maximun Service Processes* de la orden set.

Estadísticas de memoria

Las estadísticas de memoria nos pueden ayudar para determinar cómo se usa la memoria en el servidor y cuándo se hace necesario añadir más memoria. Para llegar a la pantalla de estadísticas de memoria, seleccionar Resource Utilization desde la pantalla principal de monitor. La información se presenta en tres columnas: bytes, porcentaje y uso. El porcentaje se refiere a la cantidad disponible para ese área de memoria en particular.

Debemos prestar especial atención al valor de Cache Buffers, si baja del 20% debemos añadir memoria inmediatamente. Para conseguir algo de memoria adicional hasta que añadamos efectivamente la memoria descargaremos todos los módulos que no sean estrictamente necesarios para la funcionalidad de la red, como por ejemplo los de la consola remota del servidor. También podemos utilizar la orden de consola REMOVE DOS, con ello suprimiremos el DOS de la memoria del servidor. Además, hace que la consola sea más segura porque no se pueden cargar módulos desde unidades DOS del servidor.

FIGURA 3. Las estadísticas de memoria son útiles para ver posibles carencias de memoria en el servidor.



La orden set

Hemos visto cómo utiliza nuestro servidor la memoria, podemos cambiar algunos parámetros del sistema para conseguir mejor rendimiento. Existen muchos parámetros para set, y su fórmula general es SET parámetro=valor. Por ejemplo, SET MINIMUN FILE CACHE BUFFERS=40. Básicamente los parámetros de SET se reparten en las siguientes categorías.

- Comunicaciones.
- Memoria.
- Caché de archivos.
- Caché de directorios.
- Miscelánea.

Caché de archivos

Gran parte de la eficiencia de Netware se debe al caché de datos, las tablas de archivos y su indexación. Con set podemos cambiar ciertos parámetros que nos permitirán aprovechar las mejoras en la eficiencia en sistemas que no tienen suficiente memoria para tratar la mayoría de las lecturas de disco desde la memoria caché. Los valores por omisión suelen ser adecuados, pero podemos hacer un 'ajuste fino' para que coincidan con nuestro particular entorno de red. El uso adecuado de las opciones de set puede incrementar el rendimiento del servidor y evitar tener que comprar memorias innecesariamente, pero si observamos que al utilizarlas nos pasa como a los economistas, que cuando hacen por un lado, deshacen de otro, y cuando consiguen que algo vaya bien es a costa de que otra cosa vaya muy mal, no hagamos más cábalas, hay que comprar más memoria.

Cuando las estaciones de trabajo piden al servidor entradas de disco, éste determina si puede satisfacer la petición con la información que tiene almacenada en la caché, o si necesita una lectura de disco. Tanto las lecturas como las escrituras al disco pasan por la caché; antes las escrituras, que se colocan primero en la caché y luego se planifica la escritura en disco. Podemos equilibrar las operaciones de lectura y escritura. Por ejemplo, muchas pequeñas operaciones de escritura en el disco pueden hacer caer la eficiencia de las operaciones de lectura. En este caso, lo adecuado sería establecer un retardo mayor para escribir la información desde la caché al disco con el parámetro Dirty

Disk Cache Delay Time (tiempo de retardo para las escrituras de caché modificada en disco). El valor por omisión es de 3,3 segundos. Es decir, la información de la caché es escrita en disco cada tres segundos. Para ello se paralizan todas las operaciones de lectura y se vuelca la información de la caché al disco. Podemos incrementar este valor hasta 10 segundos, pero debemos ser cuidadosos; si hubiera un corte de luz, o el sistema se colgase, perderíamos la información de los últimos 10 segundos, y en una red, eso puede ser mucha información. Con valores de retardo altos, es muy recomendable tener un sistema de alimentación de seguridad y tener activo el sistema de seguimiento de transacciones de la base de datos.

Si por el contrario en nuestra red hay muy pocas escrituras al disco, el parámetro que incrementará el rendimiento es *Maximun Concurrent Disk Cache Writes* (Máximo de escrituras concurrentes en la caché del disco). Para evitar que las operaciones de escritura consuman mucho tiempo, por defecto sólo se realizan 50 operaciones de escritura, de modo que las peticiones de lectura se atiendan rápidamente. Si por el contrario hay muchas escrituras al disco, podemos incrementar el número de escrituras al disco hasta un valor máximo de 4.000. Recordemos que durante el tiempo empleado para la escritura se inhabilitan las operaciones de lectura. Esto puede compensarse si tenemos suficiente memoria para caché de archivos, ya que aumentamos la probabilidad de que una petición de lectura sea atendida desde la caché, mientras se está escribiendo en disco, y cuanta más memoria para caché tengamos se necesitarán menos accesos al disco para lectura.

Si la utilidad monitor nos da un valor alto en Dirty Cache Buffers (alrededor del 75%) y este valor se mantiene o incrementa con el tiempo, nos indica la existencia de un problema en relación con disco y la caché; sería entonces beneficioso incrementar el valor de Maximun Concurrent Disk Cache. Si da la impresión de que el disco tarda en responder a las peticiones de lectura, debemos añadir memoria y bajar este valor.

Para modificar el sistema de caché de archivos de Netware debemos tener presentes las siguientes consideraciones :

Comprobar, mediante la utilidad monitor, que el sistema posee la memoria necesaria para gestionar eficientemente el sistema de caché.

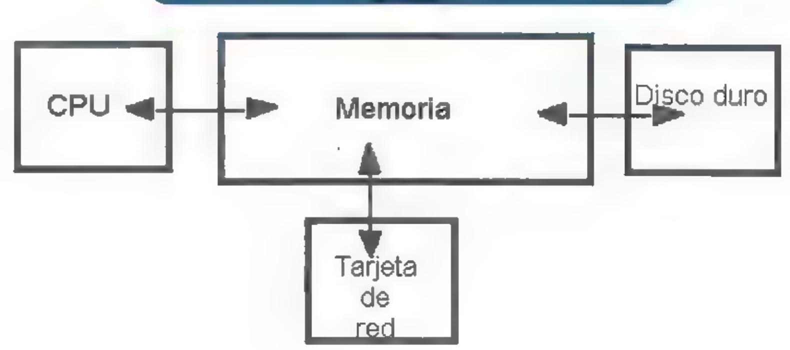
Si al cargar los NLM's se reciben mensajes de memoria insuficiente, se puede reservar más memoria para los procesos del servidor aumentando la asignación para *Minimun File Caché Buffers*, (Mínimo de Búferes para caché de archivos).

Se puede hacer que el servidor nos de mensajes de aviso cuando alcance una reserva mínima de memoria mediante el valor *Minimun File Cache Buffer Report Threshold* (Umbral para mensaje de mínimo de búferes de caché).

ra adelantada LRU. Si el tiempo de caché LRU (menos recientemente usada) está por debajo de este umbral, la lectura adelantada no tendrá lugar. El rango es de 0 segundos a una hora, por defecto 10.

Minimun File Cache Buffers=número (Mínimo de búferes para caché de archivos). Asignamos un número mínimo de búferes que siempre han de estar disponibles para caché de archivos. Esto es, una reserva de memoria que no se utilizará para otros fines. Si establecemos un valor demasiado alto, es posible que nos quedemos sin memoria para los módulos cargables y otras operaciones que ocasionalmente requieran memoria. El rango es de 20 a 1.000, por defecto el mínimo de 20.

FIGURA 4. Cuellos de botella que se producen en el servidor.



Las principales opciones que podemos establecer para caché de archivos son:

Read Ahead Enabled=On/off (Habilitar lectura adelantada). Activo por omisión. Se usa para que el sistema realice o no lecturas adelantadas en la caché de bloques que muy probablemente serán demandados en el futuro. Es muy efectivo en la lectura de archivos secuenciales.

Read Ahead LRU Sitting Time Treshold=número (Umbral de lectura adelantada LRU...)

Maximun Concurrent Disk Cache Writes=número (Máximo de escrituras simultáneas de caché al disco). Lo hemos comentado anteriormente. Asignamos un valor alto para mejorar la escritura a disco y uno bajo para mejorar la lectura. El rango es de 10 a 100, por defecto 50.

Dirty Disk Cache Delay Time=número (Tiempo de retardo para escritura de caché modificada en disco). Ya hemos comentado su uso. El rango va de 0,1 a 10 segundos. El valor por defecto es 3,3.

Minimun File Cache Buffer report Threshold=número (Umbral para mensaje de mínimo de búferes para caché). Se utiliza si deseamos ser advertidos cuando el número de búferes para caché se aproxime al mínimo especificado en el parámetro *Minimun File Cache Buffers*. Lo normal es asignarle un valor igual o menor al límite inferior definido. El rango va de 0 (no advertir) a 1.000. Su valor por omisión es 20.

Caché de directorios

El caché de directorios hace que los archivos sean localizados rápidamente, manteniendo una copia de la tabla de directorios en memoria. Es complementaria del caché de archivos y, por lo tanto, ambas cachés han de estar equilibradas.

Las principales opciones que podemos establecer son:

Directory Cache Buffer NonRefenced Delay Time=número (permanencia de entradas no referenciadas). Con esta opción especificamos el tiempo que dejaremos una entrada de directorio en caché antes de que sea sobreescrita por otra a causa de no ser utilizada. El rendimiento disminuye si reducimos este valor, ya que es menos probable que las entradas de directorio estén en caché, y si lo aumentamos aumentará el rendimiento, pero también aumentaremos los requerimientos de memoria ya que serán necesarios más búferes de directorio para mantener las entradas.

Maximun Directory Cache Buffers=número (Máximo de búferes para caché de directorio). Una vez se reserva memoria para caché de directorios, ésta no es devuelta para el caché de archivos hasta que no se reinicia el servidor. Por esta razón es bueno establecer un número máximo de búferes para caché de directorio. El rango va de 20 a 4.000. Su valor por omisión es 500.

Minimun Directory Caché Buffers=número (Mínimo de búferes para caché de directorio). Con este valor reservamos un mínimo de búferes que siempre han de estar disponibles para caché de directorio. Si es demasiado bajo se reducirá el rendimiento del acceso a los archivos. Un número demasiado alto podría ser superior al que necesita el servidor en redes pequeñas y estarían mejor aprovechados en caché para archivos. Si el servidor se muestra 'perezoso' a la hora de arrancar, es síntoma de que el valor especificado es demasiado bajo. El rango va de 10 a 2.000. Su valor por omisión es 20. Si recién arrancado el servidor el valor de Directory Cache Buffers es 100 o más debemos aumentar este valor.

Dirty Directory Cache Delay Time=número (Tiempo de retardo en la escritura de la tabla de directorios). Especifica el tiempo que el sistema retiene en memoria una petición de escritura contra la tabla de directorios. Un valor alto mejora el rendimiento, pero aumentamos el riesgo de que esta tabla se corrompa; disminuir su valor provoca el efecto contrario. El rango va de 0 a 10 segundos y su valor por defecto es 0,5. Particularmente no recomiendo jugar mucho con este parámetro.

Maximun Concurrent Directory Cache Writes=número. (Máximo de escrituras simultáneas en cache de directorio). Con este valor establecemos cuantas escrituras para búferes de directorio pueden estar pendientes antes de que las cabezas del disco duro comiencen un barrido por el disco. Con un valor alto se mejoran las operaciones de escritura y con uno bajo se mejoran las de lectura. El rango va de 5 a 50 y el valor por omisión es 10.

Directory Cache Allocation Wait Time=número (Espera para la reserva de caché de directorio). Especificamos el tiempo que ha de esperar el sistema antes de reservar un nuevo bufer para caché de directorio (recordemos que una vez reservados no se devuelven para caché de archivos). Mientras dura esta espera, las nuevas peticiones de reserva serán ignoradas. Un valor bajo puede hacer que se reserven más búferes de los necesarios durante un pico en la utilización de la red, mientras que un valor alto provocará que el servidor no pueda reservar búferes con la agilidad que requiere la carga de la red. Síntoma de ello es que la búsqueda de archivos se ralentiza en exceso cuando se hace un uso intensivo de la red.

Maximun Number of Directory Handles=número (Número máximo de gestores de directorio). Un gestor de directorio es una copia en caché de los derechos de acceso al directorio, con lo cual aumentamos la velocidad a la que el sistema decide qué podemos hacer y qué no en cada directorio. Los gestores de directorio se asignan cada vez que una conexión accede a un archivo o a un directorio. El rango va de 20 a 1.000 y su valor por defecto es 20.

Como consejo en general, si la búsqueda de directorios nos parece lenta, podemos doblar los valores *Maximun Directory Cache Buffers*, *Minimun Directory Cache Buffers*, y *Directory Cache Allocation Wait Time.*



La tecnología MMX

Emilio Postigo Rian

Un nuevo concepto ha irrumpido en el mundo de la microinformática: el de la tecnología MMX.

En efecto. INTEL ha desarrollado una extensión multimedia para sus procesadores y la ha aplicado inmediatamente sobre el conocido Pentium. A partir de este momento se habla de ordenadores equipados con procesadores Pentium MMX y muy pronto se dispondrá del Pentium Pro MMX. ¿Qué significa que el Pentium MMX sea un procesador multimedia? ¿Tal vez lleva integrada una tarjeta de sonido? No. La extensión realizada por INTEL es más profunda que eso.

Con el objetivo de acelerar la velocidad de las aplicaciones multimedia, aspecto crítico de éstas, los ingenieros de INTEL se esforzaron en definir la mejor forma de hacerlo. Estudiaron en detalle el código máquina de multitud de programas y buscaron aquellos puntos dentro de los mismos que constituían un cuello de botella y, por lo tanto, ralentizaban el funcionamiento global del sistema. El resultado de este estudio estableció que el vídeo interactivo, el audio, la realidad virtual y las aplicaciones en 3D, que son la base de aquello que denominamos multimedia, poseen mucho en común: gran cantidad de cálculos, tareas en paralelo y uso, casi exclusivo, de aritmética de enteros. A la vista de ello se decidió que los programas de estas características funcionarían mucho más deprisa si se crearan instrucciones y operandos específicos para realizar estas operaciones. De esta forma la velocidad de los procesos aumentaría porque los programas se podrían codificar mejor.

¿En qué consiste la tecnología mmx?

Como resultado de esta inteligente manera de enfocar el problema, empleando maña y no fuerza, se llegó a la definición de la tecnología MMX, basada en los siguientes puntos:

Por un lado, la llamada técnica SIMD (Single Instruction, Multiple Data o Instrucción Simple, Datos Múltiples), que permite manipular muchos operandos en paralelo con una sola instrucción.

Cuatro nuevos tipos de datos de 64 bits de longitud. El llamado ***packed byte***, que consta de 8 datos distintos de 8 bits. El ***packed word***, que son 4 datos diferentes de 16 bits. El ***packed doubleword***, dos datos de 32 bits y, por último, el ***quadword***, un único dato de 64 bits.

Ocho registros de 64 bits, llamados MM0, MM1,...MM7, constituidos por los 64 bits menos significativos de los 8 registros R0, R1... R7 del coprocesador matemático.

Y, por último, 57 nuevas instrucciones diseñadas para sacar partido de la aritmética de números enteros basándose en las características ya expuestas.

Esta ampliación de la capacidad de cálculo del procesador no constituye ningún problema para el software existente en la actualidad. Las instrucciones que se han añadido no suponen un nuevo modo de funcionamiento del procesador, sino sólo una sencilla extensión del conjunto de las mismas. INTEL garantiza que todo el software que se ejecuta sobre un Pentium seguirá funcionando en un Pentium MMX. Además, dado que se espera que esta tecnología siga desarrollándose, INTEL también se ocupará de que todo el software que se escriba para este nuevo procesador sea ejecutable en los posteriores. Los programadores, por su lado, verán afectada su labor de la misma forma que quedó modificada la programación de bajo nivel con la aparición del coprocesador aritmético: si se desea que un programa se ejecute en cualquier máquina, equipada o no con un MMX, es necesario escribir el código "multimedia" crítico dos veces: una optimizada para MMX y la otra de la manera normal, para el "vulgar" Pentium.

Las nuevas instrucciones

La mejor forma de comprender la capacidad de un nuevo procesador es, sin duda, estudiar sus instrucciones. Emplearemos este sistema para tomar contacto con las características de la tecnología MMX.

Estas instrucciones son de propósito general y se clasifican en 7 categorías: aritméticas, de comparación, de conversión, lógicas, de desplazamiento, de transferencia de datos y de estado. Lo nuevo de estas instrucciones no son ellas en sí, sino su modo de funcionamiento: las instrucciones se ejecutarán sobre más de un operando. Además los resultados finales de las operaciones podrán ajustarse de dos formas distintas: mediante truncamiento (lo normal en operaciones máquina) o mediante la llamada aritmética de saturación. Se verá este concepto a medida que analicemos las instrucciones.

Un vistazo a éstas permite apreciar que su sintaxis responde casi siempre al siguiente esquema:

El prefijo P (de *packed*, empaquetado) en todas ellas. Hace referencia a que la instrucción trabaja con varios (8,4, 2 ó 1) grupos de datos simultáneamente.

El nemónico de la instrucción correspondiente: ADD para la suma, AND para el producto lógico, etc.

Un sufijo, compuesto de las letras US *(unsigned saturation)*, para aquellas instrucciones cuyos resultados se ajusten con saturación sin lógica de signo, S (*saturation*) para resultados con saturación y lógica de signo y, por último, B, W, D y Q para indicar el tipo de datos empaquetados que emplea la instrucción.

A la vista de esto se está en condiciones de descifrar casi cualquier instrucción de las expuestas en las tablas. Por ejemplo, la instrucción PADDUSB se debe leer como "suma (ADD) simultánea (P) de ocho pares de enteros de un byte de longitud (B), aplicando al resultado la técnica de saturación sin signo (US)". De forma análoga la instrucción PADDW significa "suma (ADD) simultánea (P) de cuatro pares de enteros de 2 bytes (W) de longitud". Al no especificarse ni US ni S, los resultados se redondearán por truncamiento cuando sea necesario. Veamos ahora los grupos de instrucciones en detalle.

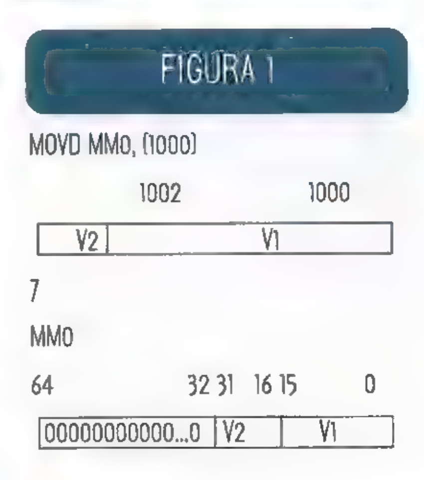
TABLA 1. Transferencia de datos.

MOV (D, Q) d, F	Mover

Intrucciones de transferencia de datos

Son las mostradas en la tabla 1.

La instrucción MOVD destino, fuente permite copiar el contenido de un registro convencional de 32 bits o el de 4 posiciones de memoria consecutivas en un registro MMX y viceversa. Está prohibida, sin embargo, la transferencia directa entre registros MMX, posiciones de memoria y registros convencionales. Si se mueve un dato de 32 bits a un registro MMX, el dato es situado en la mitad menos significativa del mismo, rellenándose la más significativa con ceros. La figura 1 muestra el resultado de pasar un valor de 32 bits, situado a partir de la dirección 1000h, al registro MM0. El contenido de las dos posiciones más significativas (direcciones 1002 y 1003) se deposita en los bits del 16 al 31, mientras que el de las direcciones 1000 y 1001 queda entre los bits 0 y 15 del registro MM0. Si el operando fuente hubiera sido el registro MM0, sólo se hubiera transferido a las posiciones de memoria el contenido de los 32 bits menos significativos de este registro, sin producirse ningún cambio en su contenido.



Por su parte, MOVQ destino, fuente trabaja sólo con registros MMX y posiciones de la memoria convencional. En su caso está admitido que los operandos des-

tino y fuente sean simultáneamente registros MMX, pero no operandos de memoria.

Tanto MOVD como MOVQ dejan inalterados los bits del registro de banderas.

Instrucciones aritméticas

Permiten sumar, restar y multiplicar grupos de números de 1, 2 ó 4 bytes de longitud, pudiéndose realizar el redondeo de los resultados por truncamiento o por saturación. El operando destino de estas instrucciones siempre ha de ser un registro MMX, mientras que el operando fuente puede ser una serie de posiciones de memoria. Mientras que las instrucciones de transferencia no ofrecían substancialmente nada nuevo, el estudio de este grupo nos pondrá en contacto directo con características MMX como son el proceso en paralelo y el ajuste de resultados por saturación. La tabla 2 contiene una lista de estas instrucciones.

TABLA 2. Instrucciones aritméticas.

PADD (B, W, DW) d, F	Suma con truncamiento
PADDS (B, W) d, F	Suma con saturación y lógica de signo
PADDUS (B, W) d, F	Suma con saturación y sin lógica de signo
PSUB (B, W, DW) d, F	Resta con truncamiento
PSUBS (B, W) d, F	Resta con saturación
PSUBUS (B, W) d, F	Resta de enteros sin signo con saturación
PMULHW d, F	Multiplicación de palabras
PMULLW d, F	Multiplicación de palabras
PMADDWD d, F	Multiplicación con suma

La figura 2 muestra dos operandos, el registro MM0 y una serie de posiciones de memoria, sobre los que se va a ejecutar la instrucción PADDW MM0, [1000]. Esta instrucción considera que

FIGURA 2

PADDW MM0, [1000]

MM0

8001	F500	FFFF	7080

1006	1004	1002	1000
8000	FFFF	5000	7000

MM0

0001	F4FF	4FFF	E080

los dos datos de 64 bits que va a sumar se dividen en 4 pares independientes de sumandos de 16 bits de longitud. El procesador efectúa las sumas depositando los resultados dentro de MM0 en las posiciones correspondientes, truncando los bits por encima del 16, como con la instrucción habitual ADD. Ahora bien, mientras que esta instrucción nos avisaba de esta situación modificando el bit CF del registro de señalizadores, la instrucción PADD, como todas las instrucciones MMX, no actualiza ninguna bandera.

Si la instrucción que se ejecuta es PADDSW MM0, [1000] o PADDUSW MM0, [1000], el procesador actúa de forma similar: en ambos casos se realizan 4 sumas de operandos de 16 bits de longitud. La diferencia de estas instrucciones con la primera, y de ellas entre sí, es la forma que se tiene de tratar los resultados que exceden la capacidad de representación del tamaño empleado. PADDUSW ajusta el resultado por saturación sin lógica de signo y PADDSW por saturación con lógica de signo.

Cuando, por ejemplo, se trabaja con datos de dos bytes de longitud sin signo, la mayor cantidad que se puede representar es 65535 (FFFFh) y la menor 0 (0000h). Si se emplea lógica de signo, la mayor cantidad representable es 32767 (7FFFh) y la menor -32768 (8000h). La técnica de redondeo por saturación consiste en asignar como resultado de una operación que excede la capacidad del operando destino el valor máximo representable, y el mínimo si el resultado de la misma es inferior a éste. Así, el resultado de PADDUSW MM0, [1000] sería FFFF FFFF FFFF E080, mientras que el de PADDSW MM0, [1000] es 8000 F4FF 4FFF 7FFF.

Las instrucciones de resta se rigen por el mismo principio.

Multiplicación

Se dispone de tres instrucciones distintas. PMULHW y PMULLW realizan cuatro multiplicaciones de operandos de 16 bits. Estas operaciones han de producir, obviamente, resultados de 32 bits, que no caben dentro de los operandos destino. La primera de ellas permite obtener los 16 bits más significativos del resultado, mientras que la segunda recoge los 16 bits menos significativos. Obtener un resultado correcto implica utilizar las dos secuencialmente.

La instrucción PMADDWD realiza cuatro multiplicaciones de números de 16 bits y realiza las sumas de los resultados por parejas. La figura 3 ilustra la forma en que esto ocurre. Esta instrucción permite codificar con facilidad algoritmos para la multiplicación de matrices y números complejos.

FIGURA 3

PMADDWD MM0, [1000]

MM0

A3	A2	A1	A0

1006	1004	1002	1000
B3	B2	B1	B0

MM0

A3*B3+A2*B2	A1*B1+A0*B0

FIGURA 4.

PCMPGTW MM0, [1000]

MM0

FFFF	0000	FFFF	7080
¿Mayor que?	¿Mayor que?	¿Mayor que?	¿Mayor que?

1006	1004	1002	1000
8000	FFFF	5000	7000

MM0

FFFF	FFFF	0000	FFFF
Sí (-1 " -32768)	Sí (0 " -1)	No (-1 " 20480)	Sí (28800 " 28672)

Comparar datos

Las instrucciones de la tabla 3 permiten determinar si los datos del operando destino son iguales (PCMPEQx) o mayores (PCMPGTx) que los datos correspondientes del operando fuente. En el caso de la condición chequeada se cumpla, los bits correspondientes del operando destino son puestos a 1, mientras que en caso contrario se rellenan con el valor 0. El operando destino de la instrucción ha de ser siempre un registro MMX y el fuente puede ser otro o bien posiciones de memoria. En el caso de la instrucción PCMPGT los datos se consideran números con signo. La figura 4 muestra cuál sería el resultado de efectuar la operación PCMPGTW MM0, [1000].

TABLA 4. Instrucciones de conversión.

PACKUSWB d, f	Convierte palabras en bytes
PACKSS (WB, DW) d, f	Convierte palabras en bytes y dobles palabras en palabras. Se emplean sobre números con signo y se aplica la técnica de la saturación
PUNPCKH (BW, WD, DQ) d, f	Convierte bytes en palabras, palabras en dobles palabras y dobles palabras en cuádruples palabras, intercalando las partes MÁS significativas de los operandos
PUNPCKL (BW, WD, DQ) d, f	Convierte bytes en palabras, palabras en dobles palabras y dobles palabras en cuádruples palabras, intercalando las partes MENOS significativas de los operandos

Instrucciones de conversión

Estas instrucciones, listadas en la tabla 4, permiten efectuar conversiones entre los nuevos tipos de datos.

TABLA 3. Instrucciones de comparación.

PCMPEQ (B, W, DW) d, f	Preguntar por IGUALDAD
PCMPGT (B, W, DW) d, f	Preguntar por MAYOR QUE

La instrucción PACKUSWB convierte palabras en bytes empleando la técnica de saturación sin lógica de signo. El valor 1000h, por ejemplo, se vería transformado en el FFh. Por su parte, la instrucción PACKSSxx convierte palabras en bytes o dobles palabras en palabras empleando la saturación con lógica de signo, de forma que los valores 1000h y 8000h pasarían a ser 7Fh y 80h respectivamente en el caso del instrucción PACKSSWB.

Ambas instrucciones emplean dos operandos, destino y fuente, debiendo ser el destino un registro MMX. La instrucción que sea toma el contenido de los dos operandos y deposita el resultado de la conversión del contenido de ambos en el operando destino. En la figura 5 se puede apreciar el efecto de esta instrucción.

Las instrucciones PUNPCKHxx y PUNPCKLxx realizan la operación inversa:

FIGURA 5

PACKSSWB MM0, [1000]

MM0

A3	A2	0000	FFFF
1006	1004	1002	1000
B3	B2	8000	7000

MM0

A3'	A2'	00	FF	B3	'B2'	80	7F

convierten bytes, palabras y dobles palabras en palabras, dobles palabras y cuádruples palabras, respectivamente. La primera de ellas toma siempre la mitad más significativa de los operandos destino y fuente y deposita en el operando destino los datos de ambos intercalándolos como se muestra en la figura 6. La instrucción PUNPCKLxx hace lo mismo pero con la parte menos significativa de los operandos. En cualquier caso, si el operando destino está inicializado con ceros, el resultado es una extensión sin signo de los datos del operando fuente.

Desplazamientos

Las instrucciones de la tabla 5 permiten efectuar desplazamientos lógicos y aritméticos sobre datos empaquetados. El operando destino es un registro MMX y el operando contador puede ser un registro MMX, un operando de memoria de 64 bits o una constante de 8 bits de longitud.

La instrucción PSLLx realiza un desplazamiento de los bits de los datos del operando destino tantos lugares como indique el operando contador. Los lugares que van quedando libres por la derecha se rellenan con ceros y los bits que "salen" por la izquierda, sencillamente, se pierden. La instrucción PSRLx efectúa la misma operación pero hacia la derecha. La primera de ellas sirve, entre otras cosas, para multiplicar rápidamente un número por una potencia de 2 (si se desplaza 1 bit hacia la izquierda se está multiplicando por 2, si se desplazan 3 bits por 8, etc.) y la segunda para realizar una división entera entre potencias de 2 (desplazar 1 bit a la derecha es lo mismo que dividir entre 2, si se desplazan 4 bits se está dividiendo entre 16).

La instrucción PSRAWx es el desplazamiento aritmético a la derecha y se diferencia de PSRLx únicamente en que los bits que van quedando libres por la izquierda no se rellenan con ceros de forma preestablecida, sino con el bit de signo (el más significativo) del operando en concreto. Esta instrucción equivale a una división entera entre 2 de números con signo. En la figura 7 tenemos los resultados de aplicar sobre MM0 las instrucciones PSRAW MM0, 2 y PSRLW MM0, 2 respectivamente.

Instrucciones lógicas

Son las mostradas en la tabla 6. En ellas el operando destino ha de ser, como siempre, un registro MMX. El operando fuente puede ser un registro MMX o un operando de memoria de 64 bits. Las operaciones lógicas se realizan bit a bit entre ambos operandos quedando modificado el operando destino. En la instrucción PANDN primero se efectúa una negación (NOT) del operando destino y, seguidamente, se ejecuta un producto lógico PAND con el fuente.

TABLA 5. Instrucciones de desplazamiento.

PSLL [W, D, Q] d, c	Desplazamiento lógico a la izquierda
PSRL [W, D, Q] d, c	Desplazamiento lógico a la derecha
PSRA [W, D] d, c	Desplazamiento aritmético a la derecha

FIGURA 6

PUNPCHKWD MM0, [1000]

MM0

A3	A2	A1	A0
1006	1004	1002	1000
B3	B2	B1	B0

MM0

A3	B3	A2	B2

La instrucción emms

Esta instrucción se emplea para finalizar las operaciones MMX dejando el coprocesador listo para su empleo. ¿Qué significa esto?

Como ya se vio, los nuevos registros MMX consisten en los 64 bits menos significativos de los registros del co-

FIGURA 7

MM0 inicial

FFFF	8000	0000	F00F

PSRLW MM0, 4

MM0 final

0FFF	0800	0000	0F00

PSRAW MM0, 4

MM0 final

FFFF	F800	0000	FF00

procesador. Esto se traduce en que no es posible utilizar simultáneamente instrucciones MMX e instrucciones con datos en coma flotante.

Los registros del coprocesador poseen una situación precisa dentro de éste, pero son manejados por las instrucciones de coma flotante como si fueran una pila con una posición inicial variable. El valor que indica cuál es el primer elemento de la misma en un instante determinado se almacena en tres bits de un registro llamado palabra de estado y se denomina TOS (*Top Of Stack*). Las instrucciones en coma flotante manejan estos registros identificándolos por su posición con respecto al inicial, por lo que se suele hablar de ST ó ST(0), ST(1), ...ST(7). Además, los registros del coprocesador que poseen información útil se distinguen por el contenido de ciertos bits de la llamada palabra de marca (TAG). Este registro posee 2 bits para cada registro del coprocesador. Si el valor almacenado en ellos es 11, significa que el registro está "vacío", mientras que si almacena 00 el registro correspondiente está siendo utilizado por alguna instrucción. La figura 8 muestra una situación posible: los registros R5, R6 y R7 poseen información útil y los bits correspondientes de la palabra de marca almacenan el valor 00; los bits de la palabra de estado correspondientes al TOS guardan el valor 5, por lo que R5 ha de ser manipulado a través de una instrucción por medio del operando ST o ST(0). Cuando se trabaja con el coprocesador es imprescindible que los valores mencionados se conserven, de manera que las operaciones que se ejecuten sean correctas.

TABLA 6. Instrucciones lógicas.

PAND d, f	Operación AND
PANDN d, f	Operación AND NOT
POR d, f	Operación OR
PXOR d, f	Operación XOR

TABLA 7. Estado.

EMMS	Habilita el empleo fiable del coprocesador

Cuando se ejecuta un instrucción MMX se producen varios hechos que afectan directamente el estado del coprocesador. Al margen de que la mantisa de sus registros se ocupa con datos enteros, por lo que ya no son utilizables por el mismo, sus bits de signo y exponente se rellenan automáticamente con el valor 1. Además el registro de marca (TAG) del coprocesador se rellena con ceros en todas sus posiciones, indicando que los registros del coprocesador están ocupados, y el puntero TOS se inicializa con 000. Por este motivo la ejecución de una instrucción MMX en mitad de un proceso de coma flotante tendría como resultado la pérdida de los datos.

El objeto de la instrucción EMMS es evitar que una instrucción en coma flotante se encuentre con el coprocesador bloqueado. Su ejecución restaura el contenido de la palabra de marca de manera que todos los registros sean considerados "vacíos". Si esta instrucción no es empleada y se intenta ejecutar un fragmento de código

FIGURA 8

TAG

R7	R6	R5	R4	R3	R2	R1	R0
00	00	00	11	11	11	11	11

Palabra de estado

		1					

	79 — 64	63 — 0	
R7		LLENO	ST(2)
R6		LLENO	ST(1)
R5		LLENO	ST(0)
R4		VACÍO	
R3		VACÍO	
R2		VACÍO	
R1		VACÍO	
R0		VACÍO	

que emplea el coprocesador pueden generarse excepciones y errores, el menor de los cuales es que el resultado de la operación sea incorrecto. Esta instrucción es imprescindible después de emplear instrucciones MMX y antes de conmutar de tarea trabajando con un sistema operativo multiusuario.

NUEVOS FORMATOS

Packed Byte: 8 bytes contiguos en memoria

63 56 55 48 47 40 39 32 31 24 23 16 15 8 7 0

Packed Word: 4 palabras contiguas en memoria

63 48 47 32 31 16 15 0

Packed Double-Word: 2 dobles-palabras contiguas en memoria

63 32 31 0

Los nuevos registros

79 64 63 0

R7		MM7
R6		MM6
R5		MM5
R4		MM4
R3		MM3
R2		MM2
R1		MM1
R0		MM0

Precauciones

A la vista de lo anterior, es recomendable aislar el código MMX del código en coma flotante, finalizando siempre el del primer tipo con la instrucción EMMS. Por su parte, el código en coma flotante debe finalizar dejando la pila vacía. No se debe almacenar resultados intermedios dentro de estos registros.

Detectar la tecnología MMX

Al mencionarse la manera en que la aparición de esta tecnología iba a afectar la labor de los desarrolladores, se señaló que éstos deberían codificar las rutinas críticas por duplicado, de forma que el programa pudiera sacar el máximo partido del procesador presente cuando éste fuera un Pentium MMX. Esto implica que debe existir un sistema que permita a una aplicación detectar la presencia de esta tecnología, y la clave se encuentra en la instrucción CPUID (CPU IDentification).

Esta instrucción aparece en los procesadores INTEL con el Pentium y está pensada para que el software pueda conocer el modelo del procesador en el que se está ejecutando. El programador debe situar en el registro de 32 bits EAX un valor determinado, de manera que la instrucción "sepa" qué tipo de información debe devolver. Por ejemplo, un valor de 00000000 en EAX provocará que la instrucción CPUID devuelva, en el caso de un Pentium INTEL, la cadena de 12 caracteres "GenuineIntel" distribuida a lo largo de los registros EBX, EDX y ECX.

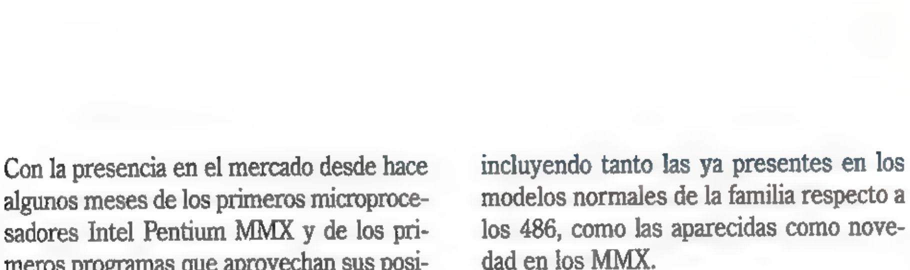
Pero para el tema que nos ocupa el valor que debe ser depositado en EAX es 0000001. Al ejecutarse en estas circunstancias CPUID, se obtendrá un valor en EDX que nos permitirá saber si el procesador posee extensión multimedia o no. Si el bit 23 de EDX es 1, el procesador es un auténtico Pentium MMX. El sencillo programa que se adjunta ilustra la forma en que se debe proceder. Si este programa se ejecuta en un procesador anterior al Pentium, el no reconocimiento de la instrucción CPUID provocará la excepción 6 (código de instrucción inválido).

```
.586
Datos SEGMENT PARA USE16 'DATA'
    NoHayMMX DB 'No se encontró la tecnología MMX$'
    HayMMX   DB 'Tecnología MMX presente$'
Datos ENDS
Pila SEGMENT PARA STACK USE16 'STACK'
    DB 1024 DUP (?)
Pila ENDS
Codigo SEGMENT PARA USE16 'CODE'
    ASSUME CS:Codigo, DS:Datos, SS:Pila
main:
            mov ax, Datos
            mov ds, ax
            mov eax, 1
            cpuid
            test edx, 00800000h
            jnz MMX_Encontrada
            mov dx, offset NoHayMMX
            jmp @F
MMX_Encontrada:
            mov dx, offset HayMMX
@@:
            mov ah, 09h
            int 21h
                mov ax, Datos
            mov ds, ax
                mov ah, 4ch
            int 21h
Codigo ENDS
    END main
```

MMX, hardware y programación

Jorge Figueroa





Ahora que los microprocesadores MMX ya empiezan a ser una realidad, vamos a desvelar qué mejoras hardware incluyen estos micros y qué ventajas reales proporcionan a los programadores en las diversas áreas de la programación multimedia

Con la presencia en el mercado desde hace algunos meses de los primeros microprocesadores Intel Pentium MMX y de los primeros programas que aprovechan sus posibilidades (principalmente juegos), es hora de que el lector conozca realmente las mejoras que incorporan estas CPUs, incluyendo en ello tanto en lo que a hardware se refiere, como a las instrucciones MMX en sí mismas y las ventajas reales que ofrecen éstas a los programadores a la hora de escribir aplicaciones que usen lo actualmente conocido como elementos multimedia, incluyendo en esto tanto sonido, gráficos, vídeo, comunicaciones y todo lo relacionado.

El hardware de un Pentium MMX

Como todos sabemos, las extensiones MMX son un conjunto de nuevas instrucciones que se han añadido a los Pentium orientadas a mejorar la velocidad de las aplicaciones multimedia, pero ¿poseen más mejoras estos micros?

La realidad de estos micros es que las mejoras que se les han aplicado son bastantes, puesto que si no fuera así, el aumento de rendimiento y la aceptación que hubiesen tenido sería muy pobre.

A continuación se van a listar las principales características de los Pentiums MMX, incluyendo tanto las ya presentes en los modelos normales de la familia respecto a los 486, como las aparecidas como novedad en los MMX.

Características generales de la familia pentium

1.- Bus de 64 bits real. Como era de esperar, junto con todo un set de instrucciones cuya característica principal es la de que pueden operar con registros de 64 bits, los micros poseen un Bus de datos de 64 bits, con lo que las transferencias de información se aceleran por dos.

2.- Dado que las CPUs cada vez tienen que manejar mayores cantidades de datos, el micro incluye características para el control de la integridad de datos.

3.- La ejecución de instrucciones que operan con enteros se realiza con dos unidades de procesamiento paralelas, con lo que se consigue la ejecución de hasta dos instrucciones simultáneas por ciclo de reloj.

4.- La unidad de Punto Flotante, el comúnmente llamado coprocesador

matemático, puede procesar hasta dos instrucciones en paralelo gracias a su diseño tipo *pipeline*. A parte de ello, está muy optimizado, pudiendo realizar una suma o multiplicación de reales en un sólo ciclo de reloj.

5.- Parte de las instrucciones anteriormente escritas en forma de microcódigo están ahora "hardwarificadas" para requerir de esta forma un menor número de ciclos para ser ejecutadas.

6.- Poseen un sistema de paginación que les da mayor capacidad de direccionamiento y que soporta memoria virtual en disco.

7.- Incorporan un sistema de predicción de saltos que acelera las instrucciones de salto que han sido ejecutadas hace poco, lo cual se nota principalmente en bucles y algoritmos complejos.

8.- Inclusión de mejoras en las opciones de monitorización de su funcionamiento.

9.- Sistema de gestión de interrupciones virtual (VIF & VIP).

10.- Modo de funcionamiento del micro y Controlador de Bus preparados para poder ser utilizados en sistemas basados en plataformas bi-procesador.

11.- Sistema de gestión de Consumo de energía incluido.

12.- Detección de errores hardware en los dispositivos conectados mediante paridad.

13.- La instrucción CPUID devuelve toda la información necesaria para poder identificar el microprocesador.

14.- Los modelos Pentium clásicos poseen 2 bloques de 8 Kb cada uno de memoria caché level 1 para código y datos respectivamente. Este sistema mejora el rendimiento respecto al búfer único de código+datos que incluían los 486 (que además era de 8Kb para ambos).

Novedades y mejoras del modelo MMX

1.-Set completo de instrucciones MMX incluido, basado en la tecnología SIMD (Single Instruction, Multiple Data). Para más información al respecto ver artículo sobre el tema aparecido en el presente número de la revista.

2.- Dos bloques de memoria caché level 1 de 16 Kbytes cada uno separados para código y datos, lo cual, aunque no parezca mucho respecto a los 8+8 de los Pentium Normal, puede aumentar considerablemente el rendimiento global del sistema.

3.- El Sistema de predicción de saltos se ha mejorado, con lo que la ejecución de bucles se acelera notablemente. Se basa en una especie de caché de saltos donde se guardan los últimos saltos realizados por las últimas instrucciones ejecutadas de salto condicional e incondicional.

4.- Mejoramiento del *Pipeline*.

5.- Se ha mejorado la gestión de los accesos a memoria, separando los búferes de escritura de cada unidad de proceso para evitar cuellos de botella cuando se ejecutan dos instrucciones en paralelo que acceden a direcciones de memoria coincidentes.

6.- Mejoras en la velocidad de ejecución de las instrucciones más criticas del Pentium clásico.

Ventajas prácticas del Pentium MMX

Según fuentes de la propia Intel, el microprocesador Pentium MMX ofrece en las aplicaciones normales no diseñadas específicamente para él, entre un 10 y un 20% más de rendimiento global, lo cual deriva principalmente de la incorporación de una mayor caché tipo *level* 1 y por la unidad de predicción de saltos mejorada que se le ha incluido.

En cuanto a las aplicaciones de tipo multimedia se refiere, que además es lo que más nos interesa realmente, el microprocesador puede dar hasta un 60% más de rendimiento global, aunque dicho aumento depende también del tipo de aplicación de que se trate.

Para tener una referencia general, en el test iCOMP(R) Index 2.0, un Pentium MMX a 200 Mhz. obtiene un iCOMP de 182 mientras que un Pentium normal a la misma velocidad de reloj, da un resultado de 142. Dados los números, se puede observar la diferencia de rendimiento

Utilidad del set MMX

Aunque el lector conozca ya cuáles son las instrucciones MMX y para qué sirven, su utilidad a simple vista como instrucciones multimedia podría quedar poco clara y por eso este aspecto se va a comentar en lo que queda de artículo.

Como introducción al tema, en el listado 1 se muestra un simple procedimiento ensamblador que nos permite saber si el sistema en el que se ejecuta es un procesador modelo MMX. Dado que usa la instrucción CPUID, no se puede ejecutar en modelos inferiores a Pentium. En caso de detectarse las extensiones, devuelve en EAX el valor 0 y en caso contrario EAX será -1.

A continuación se van a dar ejemplos de referencia acerca de técnicas y algoritmos de las principales áreas de programación consideradas multimedia que se benefician enormemente de la presencia de estas nuevas instrucciones.

LISTADO 1. Detección del MMX.

```
DETECTA_MMX proc
 mov eax,1
 cpuid
 test edx,00800000h
 jnz MMX_DETECTADA
 mov eax,-1
 ret
 MMX_DETECTADA:
 sub eax,eax
 ret
DETECTA_MMX endp
```

Gráficos 2D

En el apartado de gráficos 2D, podemos decir que las mejoras que obtenemos son las más fáciles de entender, ya que tan sólo la inclusión de las instrucción MOVQ destinada a mover bloques de datos de 64 bits entre registros MMX y la memoria hace que la velocidad en volcados de fragmentos de búfers gráficos sea el doble.

Otro ejemplo muy claro es el apartado de los *sprites* transparentes, ya que podemos decir que gracias a la inclusión de instrucciones tales como PCMPEQB, la detección de las partes transparentes de los gráficos se realiza muy rápidamente, puesto que se pueden procesar las transparencias de hasta 8 píxeles en paralelo (en caso de usar gráficos de 8-bits/pixel), por lo que se acelera notablemente la gestión de muchos gráficos, como es el caso de los arcades 2D (hasta 4 veces), donde normalmente el número de *sprites* en pantalla es muy elevado.

Otros aspectos de las 2D menos aparentes que se pueden mejorar son también por ejemplo los relacionados con los fractales, puesto que los algoritmos usados para generar fractales 2D en tiempo real, como los utilizados por ejemplo en la *naturalización* de decorados (ríos, árboles...), se optimizan hasta un 40-50%, gracias a las capacidades de las MMX.

En el Listado 2 podemos ver un ejemplo de código MMX aplicado al dibujado de *sprites* transparentes.

LISTADO 2. Dibujo de un *sprite*.

```
;-------------------------------
; Fragmento de una rutina para
; el dibujado de un sprite.
; Bucle principal
;-------------------------------
LoopTop:
 add ebx, 8
 add ecx, 8
 movq mm0, [eax]
 pxor mm7, mm7
 movq mm1, [ecx]
 pcmpeqb mm7, mm0
 add eax, 8
 pand mm7, mm1
 add ebp, 8
 por mm7, mm0
 movq [ebx], mm1
 movq [ecx], mm7
 cmp ebp, edi
jne LoopTop
```

Gráficos 3D

En el apartado de gráficos en 3D, pese a la proliferación actual de las placas gráficas 3D que hacen innecesario el uso de la CPU para cálculos 3D, las intrucciones MMX consiguen que el procesamiento se haga más rápido a la hora de realizar efectos de Gouraurd, Z-Búffer, filtros bilineales, mapeado de texturas, Alpha Blending y otros relacionados o parecidos, donde las mejoras de rendimiento obtenidas pueden ser de hasta un 100% o un 200% en caso de que estemos usando color real (24 bits) o de 16 bits. Por otra parte, en el caso de usar color de 8 bits, la mejora es menor, ya que se aumenta el rendimiento sólo un 30 por ciento.

Estas mejoras en 3D son posibles básicamente a las siguientes propiedades MMX:

1.- Procesado en paralelo (SIMD).

2.- Instrucción MOVQ capaz de mover bloques de 64 bits. Lo cual es importante cuando un sólo píxel puede ocupar 24 bits.

3.- Posibilidad de realizar operaciones de saturación, que facilitan los cálculos en modo color real.

4.- Posibilidad de realizar comparaciones de paquetes de datos con PCMP, muy útil en algoritmos 3D.

Sonido

En el área de la programación de sonido, la MMX también beneficia a los algoritmos más comunes, como son los efectos de eco, el algoritmo de sonido MPEG1, el G.728 para la codificación/decodificación de voz y diversos filtros digitales comúnmente usados, como son el de Levinson-Durbin y el de Schur Weiner.

Por ejemplo, el Algoritmo G.728 optimizado con código MMX puede funcionar entre un 100% y un 200% más rápido que una versión normal para Pentium, lo cual es de agradecer sobre todo si tenemos en cuenta que muchos de estos algoritmos están diseñados para ser aplicados en tiempo real durante la reproducción. Y por lo tanto, es de vital importancia que consuman el menor número de ciclos posible por segundo para no saturar el procesador.

Comunicaciones

Por comunicaciones nos referimos principalmente a los diversos algoritmos de filtrado de señal usados en las comunicaciones por módem. Ejemplos conocidos son el *<Passband Echo Canceller>*,el *<Baseband Echo Cancelation>*, el *<1/3 Equalizer>* y el *<2/3 T Spaced Equalier>* entre otros.

En este tipo de programación, las ventajas que proporciona el set MMX son debidas en casi todos los casos a las instrucciones de suma y multiplicación múltiples (como es PMADDWD), así como las de expansión de datos (como PUNPCKDQ).

Todo ello permite que se pueda operar sobre *arrays* de datos (cadenas) consecutivos de forma paralela con la consecuente optimización de los bucles en cuanto a ciclos totales necesarios.

En el listado 3 podemos ver un ejemplo de código MMX aplicado al algoritmo *<Passband Echo Canceller>*.

LISTADO 3

```
;------------------------------------------------
; Fragmento de código MMX para el procesado del
; algoritmo "Passband Echo Canceller"
;------------------------------------------------
mov   rtmp, [dl_+qcount*4-16*LONGITUD]
movq  dl0, [oldl+qcount-4*LONGITUD]
movdf [dl_+qcount*4-6*LONGITUD], dl1
movq  [dl_+qcount*4-16*LONGITUD], dl0
movq  dl4, dl1
psllq dl4, 48
psrlq dl0, 16
por   dl4, dl0
psrlq dl0, 16
psllq dl1, 16
movdf [dl_+qcount*4+8*LONGITUD], dl0
movq  [dl_+qcount*4+4*LONGITUD], dl4
psrlq dl0, 16
por   dl1, dl0
movq  [dl_+qcount*4+12*LONGITUD], dl1
movq  dl1, [oldl+qcount]
mov   rtmp, [dQ_+qcount*4]
movdf [dQ_+qcount*4+10*LONGITUD], dQ1
movq  dQ0, [oldQ+qcount]
movq  dQ4, dQ1
movq  [dQ_+qcount*4], dQ0
psllq dQ4, 48
psrlq dQ0, 16
por   dQ4, dQ0
psrlq dQ0, 16
movq  [dQ_+qcount*4+4*LONGITUD], dQ4
psllq dQ1, 16
movdf [dQ_+qcount*4+8*LONGITUD], dQ0
psrlq dQ0, 16
por   dQ1 dQ0
movq  [dQ_+qcount*4+12*LONGITUD], dQ1
movq  dQ1, [oldQ+qcount]
```

Efectos digitales

En el apartado de efectos digitales, los incrementos de velocidad se perciben también en todos los algoritmos en general, aunque hay algunos que se aceleran especialmente dadas sus características.

El ejemplo más claro y sencillo sería el dibujado de un *sprite* en tamaño doble(x2). En este caso, la presencia de las instrucciones PUNPCKHBW y PUNPCKLBW, que permiten la conversión de bytes simples a dobles de forma rapidísima y en bloques de 4 bytes en paralelo por instrucción ejecutada, hacen que el tiempo de proceso necesario para duplicar el tamaño X del gráfico de color 8 bits se reduzca hasta en 4 veces. En caso de usar modos de mayor profundidad de color, el efecto puede utilizarse igual pero con menor paralelismo de procesado (menos píxeles duplicados por instrucción SIMD ejecutada).

En el listado 4 podemos ver el ejemplo de un procedimiento ensamblador optimizado con instrucciones MMX creado para el dibujado de imágenes x2.

Vídeo

El mundo de la reproducción de vídeo digital y todo lo relacionado con él, incluyendo filtros, conversiones de color y demás, siempre ha sido un aspecto muy limitado de los PCs dado lo costoso que resulta su procesado, tanto por la cantidad de información que hay por procesar por segundo como por la complejidad de los propios algoritmos que intervienen.

En este caso, para no ser menos, los principales algoritmos empleados comúnmente, como es el *<Motion Compensation>* usado en el MPEG 1, los filtros de suavizado de imagen y otros, sufren considerables aumentos de rendimiento al diseñarse con la ayuda de la MMX, pudiendo ahora realizar reproducciones de calidad de vídeo a pantalla completa.

Conclusiones para el programador

Como conclusión global, podemos decir que una vez revisadas las nuevas funcionalidades que aporta la MMX para programar, podemos deducir los siguientes aspectos:

1.- Gracias a los siete registros de 64 bits de que disponemos ahora, a los cuales podemos acceder a nivel de bytes, palabras o "doblespalabras" de forma individual, la creación de rutinas complejas que requieren muchos parámetros temporales se ve muy simplificada, ya que se hace innecesario el uso de variables de memoria y/o de la pila. Otra consecuencia es que los tiempos de ejecución se ven reducidos significativamente ya que, como sabemos, la memoria RAM es de muy lento acceso.

2.- Las instrucciones capaces de realizar operaciones aritméticas simples (PADDW, PADDSW, PADDUSW, PADDSW...) con varios datos simultáneamente y tanto por méto-

LISTADO 4

```
;-------------------------------------
-
; Ejemplo de rutina de dibujado de gráfico
; en tamaño x2 usando MMX.
; -----------------------------------
-
TITLE IMAGEN_2X
.nolist
INCLUDE iammx.inc
.list
.586
.model FLAT
; --------------------
; SEGMENTO DE DATOS
; --------------------
.data
x_half DWORD 0H
;--------------------
, SEGMENTO DE CODIGO
; -------------------
.code
; BLOQUE_MEM: Puntero al inicion de la superficie
; TAM_X: Anchura del gráfico
; TAM_Y: Altura del gráfico
IMAGEN_2X proc near C uses eax ebx ecx edx edi
esi,
   BLOQUE_MEM: PTR BYTE,
   TAM_X: DWORD,
   TAM_Y: DWORD
  mov   edi, TAM_X
  mov   edx, TAM_X
  mov   eax, edi
  shr   edx, 1
  mov   esi, edi
  imul  eax, TAM_Y
  mov   ebx, eax
  add   eax,BLOQUE_MEM
  shr   ebx, 1
  sub   eax, edi
  mov   x_half, edx
  add   ebx,BLOQUE_MEM
  mov   ecx, eax
  sub   ebx, x_half
  shl   esi, 1
  sub   ecx, edi
 loopstart:
  movq  mm0, [ebx - 32]
movq  mm1, mm0
mov   edx, [eax - 64]
sub   ebx, 32
mov   edx, [eax - 32]
sub   eax, 64
punpcklbw mm0, mm0
movq  mm2, [ebx + 8]
punpckhbw mm1, mm1
movq  [eax + edi], mm0
movq  [eax + edi + 8], mm1
movq  mm3, mm2
movq  [eax], mm0
punpcklbw mm2, mm2
movq  [eax + 8], mm1
movq  [eax + 16], mm2
punpckhbw mm3, mm3
movq  mm4, [ebx + 16]
movq  [eax + 24], mm3
movq  mm5, mm4
movq  [eax + edi + 16], mm2
punpcklbw mm4, mm4
movq  [eax + edi + 24], mm3
punpckhbw mm5, mm5
movq  [eax + 32], mm4
movq  mm6, [ebx + 24]
movq  [eax + 40], mm5
movq  mm7, mm6
movq  [eax + edi + 32], mm4
punpcklbw mm6, mm6
movq  [eax + edi + 40], mm5
punpckhbw mm7, mm7
movq  [eax + 48], mm6
movq  [eax + 56], mm7
movq  [eax + edi + 48], mm6
movq  [eax + edi + 56], mm7
cmp   ecx, eax
jne loopstart
sub   ecx, esi
sub   eax, edi
sub   ebx, x_half
cmp   eax,BLOQUE_MEM
ja loopstart
emms
ret
IMAGEN_2X ENDP
END
```

do de saturación como por truncamiento, las convierten en útiles en casi cualquier tipo de rutina que podamos crear, puesto que casi siempre están presentes arrays de datos en cualquier tipo de rutina que necesitan ser procesados de alguna forma con operaciones matemáticas sistemáticas.

3.- La presencia de la instrucción matemática PMADDWD se hace muy útil en el caso de algoritmos complejos, como cuando necesitamos realizar cálculos de matrices, lo que es muy frecuente en casos de rutinas de tipo DSP, como son el <Efficient Vector>, la transposición de matrices, la FFT (Fast Fourier Transform) o el cálculo del *Dot Product* de dos vectores.

El futuro de la MMX

Pese a que la MMX acaba de aparecer, podemos ya hablar con certeza de cuál va a ser su aceptación y su futuro. Dado que hay un modelo Pentium MMX en el mercado, puesto que Intel va a presentar en breve un Pentium Pro con MMX (llamado Pentium II) y sabiendo también que el próximo modelo de micro de dicha casa ya las incorporará de entrada, podemos decir que no se trata de una alternativa o complemento, sino de un elemento que todos los micros futuros poseerán. Además, por si esto fuera poco, otras empresas como AMD y Cyrix, ya han anunciado la próxima aparición de modelos compatibles MMX de sus respectivas casas. Todo ello, sumado al beneficio real que aporta dicha tecnología en el rendimiento global del sistema, hace que dicha tecnología tenga el futuro asegurado. Por si algún lector desconfía aún del futuro de la MMX, sólo decirle que Intel ha anunciado que saldrá en 1998 una versión nueva del Pentium Pro llamada por ahora **KATMAI**, que incluirá la MMX 2, una versión nueva (ampliada) de dicho set de instrucciones.



Propiedades y eventos

Enrique de la Lastra

Delphi permite la definición de nuevas propiedades de los componentes, de forma que sea posible que los componentes creados sean configurables en tiempo de diseño (y, por supuesto, en tiempo de ejecución) por el usuario del componente, es decir, por el creador de aplicaciones.

Es necesario tener siempre presente las dos facetas de la programación en Delphi. Por un lado, el creador de aplicaciones, el cual, mediante los componentes que incorpora el compilador, diseña una serie de formularios y una *navegación* entre ellos y proporciona la funcionalidad deseada a cada uno a través del lenguaje de programación. Hasta aquí, el programador no necesita tener ningún conocimiento de programación orientada a objetos (OOP: *Object Oriented Programming*), ni de cómo es la estructura de los componentes. Tan sólo necesita saber programar en un lenguaje de alto nivel y tener una cierta habilidad de diseño de modo que la interfaz de la aplicación sea óptima para el usuario de la misma. Sin embargo, existe otro nivel de programación: la programación *orientada a componentes*. Este tipo de programación, introducida en el artículo del mes anterior, se basa en conceptos de OOP y en otros propios de Delphi y se utiliza para crear componentes Delphi. En este nivel, el programador debe conocer íntimamente la estructura de los componentes y manejar con soltura la OOP que proporciona Delphi. La diferencia más importante entre estos dos "niveles" de programación es el *destinatario* de lo creado.

FIGURA 1. Cambio de la propiedad caption de un label

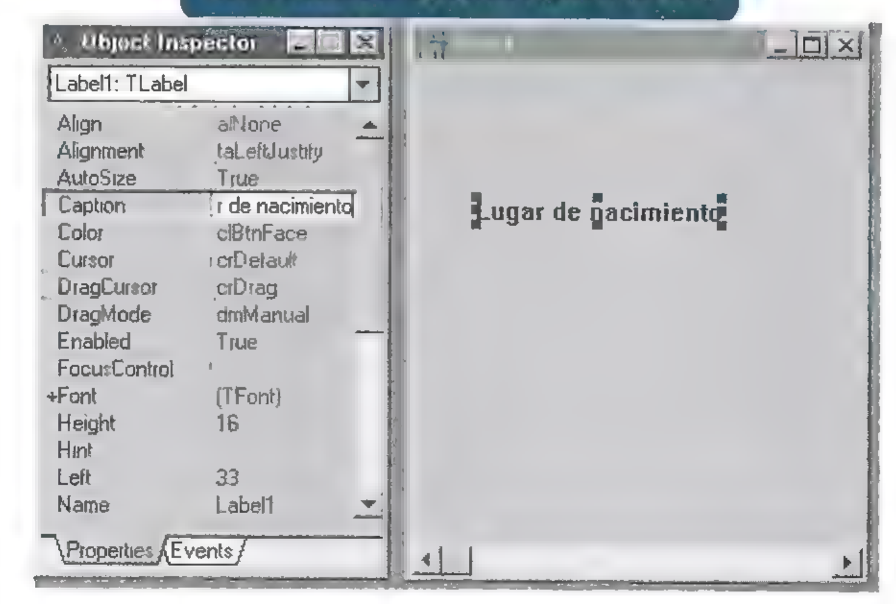
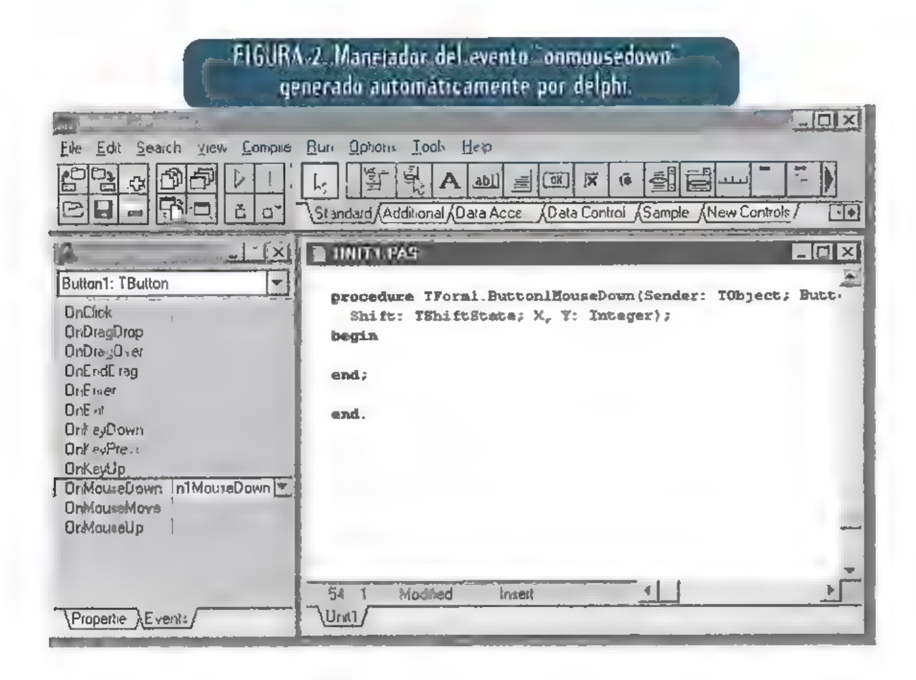


FIGURA 2. Manejador del evento "onmousedown" generado automáticamente por delphi.



El programador de aplicaciones hace uso de los componentes y tiene como destinatario al usuario final de la aplicación. El programador de componentes tiene como destinatario al desarrollador, quien integrará esos componentes en sus aplicaciones. En este sentido, cuando se crea un componente es preciso tener presente que quien lo va a utilizar va a ser un desarrollador de software y, por tanto, se le proporcionará acceso a las propiedades del mismo que permiten "hacer algo" y se le "esconderá" todo aquello que no resulte significativo.

La forma de conseguirlo es mediante las *propiedades*. Si seleccionamos un componente de la paleta de componentes, por ejemplo, un "Label" (Etiqueta) y lo pinchamos en un formulario, aparecerá inmediatamente dicho "Label" y se mostrarán en el *Object Inspector* (Inspector de Objetos) todas sus propiedades y eventos. La propiedad "Caption" (Título) permite que escribamos el texto que queremos que aparezca en tiempo de ejecución, con la facilidad adicional de que podemos observarlo en tiempo de diseño. La propiedad "Left" (Izquierda) informa de la posición del "Label" respecto a la parte izquierda del formulario y se modifica tan sólo reescribiendo un nuevo valor para la misma. Esta forma de trabajar resulta tremendamente cómoda y fácil para los desarrolladores. Pero todo lo que resulta muy sencillo de utilizar responde a una ardua labor de programación que se realiza de forma "transparente" al usuario. Al cambiar la propiedad "Caption", para que en lugar de "Label1" aparezca por ejemplo, "Lugar de nacimiento" (FIGURA 1) lo que a nosotros nos parece inmediato, no es sino el resultado de una serie de acciones internas que realizan lo siguiente: como se ha modificado una propiedad se varía una variable interna para que almacene el nuevo valor, se comprueba si hay una función a la que llamar y si es así, se utiliza (en este caso, deberá borrar el texto que aparece en pantalla y repintar el nuevo texto) y, por último, se varía el fichero *.dfm para que guarde el nuevo valor.

Propiedades

Recordamos ahora la definición de propiedad que quedó enunciada en el artículo del mes pasado: "atributo de una clase que define las acciones a tomar cuando se lee o escribe dicho atributo". Éste será nuestro trabajo cuando queramos crear una propiedad: definir qué es lo que queremos que se produzca en el componente cuando se escribe un nuevo valor en el *Object Inspector* o se cambia en tiempo de ejecución, así como cuando se lee el valor actual (en tiempo de ejecución generalmente). Una vez definidos estos aspectos, habrá que proporcionar las funciones para que aquéllos sean aplicados. Para declarar una propiedad como tal, se utiliza la palabra reservada "property" (propiedad) a la que seguirá el nombre que queremos dar a dicha propiedad (como regla general, deberá ser explicativo de su contenido). Después, irá el tipo de propiedad, de igual manera a cómo se declara una variable. Detrás, debemos escribir la palabra reservada "read" (leer), seguida de la función a la que hay que llamar cuando el usuario del componente lea el valor de dicha propiedad y la palabra "write" (escribir), seguida de la función que se llama en la escritura de la propiedad. Ambas funciones pueden ser "nulas", es decir, al leer o escribir la propiedad no se hará nada más que modificar una variable interna (que, posteriormente, podrá ser utilizada en otras partes del código del componente).

Si una propiedad no debe ser modificable por el usuario se implementará de "sólo lectura", lo cual se consigue especificando sólo la parte "read" y eliminando la parte "write" de la propiedad. De igual manera, si queremos que sea de sólo escritura, es decir, que se pueda cambiar su valor pero no recuperarlo, se escribirá sólo la parte "write". Esta última opción interesa con propiedades que sólo puedan leer el código interno del componente como, por ejemplo, una propiedad "password" (contraseña).

A continuación, vemos las diferentes opciones explicadas con un ejemplo. Supongamos una propiedad "Prueba" de tipo *string* (cadena de caracteres).

1. Propiedad de lectura y escritura:

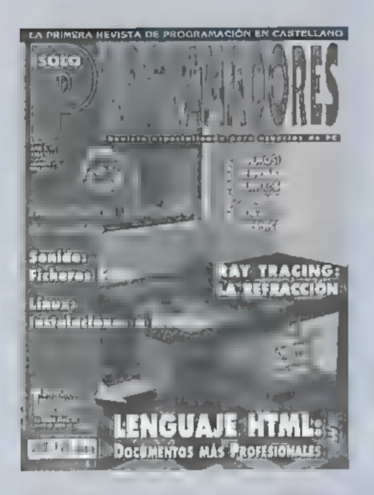
1.1. Si en lectura y escritura llamamos a una función:

Property Prueba: string read GetPrueba write SetPrueba;

números atrasados

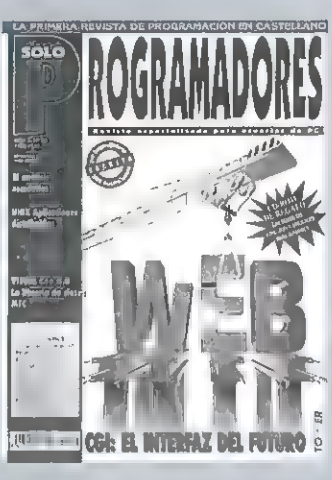
PROGRAMADORES

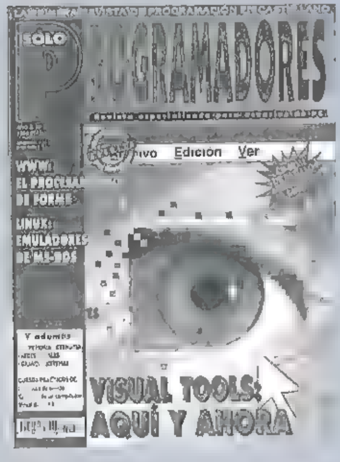
completa ya tu colección

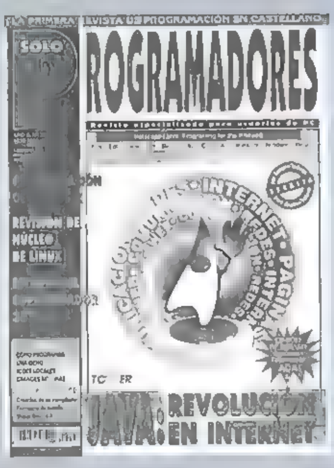








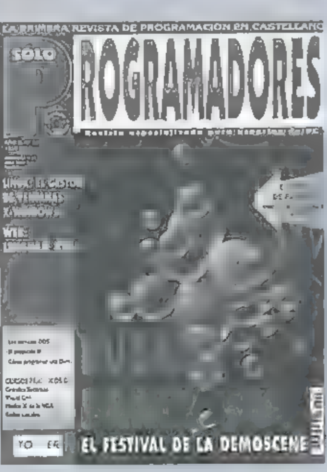




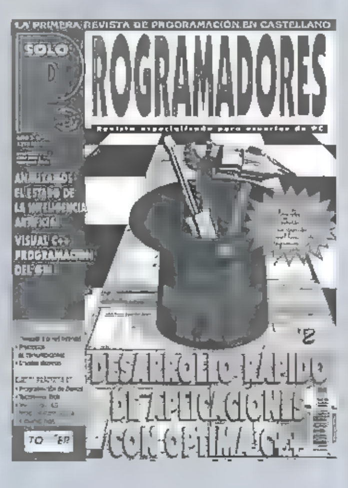












1.2. Si en lectura no llamamos a ninguna función, pero sí en escritura:

Property Prueba: String read FPrueba write SetPrueba;

Por convenio, cuando se especifica la función que se ha de llamar en lectura, se utiliza la palabra "Get" (Obtener) seguida del nombre de la propiedad, y en escritura se emplea la palabra "Set" (Fijar). Si no se llama a ninguna función, también por convenio se utiliza la letra "F" seguida del nombre de la propiedad (FPrueba en nuestro caso).

2. Propiedad de sólo lectura:

Property Prueba: string read GetPrueba;

o

Property Prueba: string read FPrueba;

3. Propiedad de sólo escritura:

Property Prueba: string write SetPrueba;

o

Property Prueba: string write FPrueba;

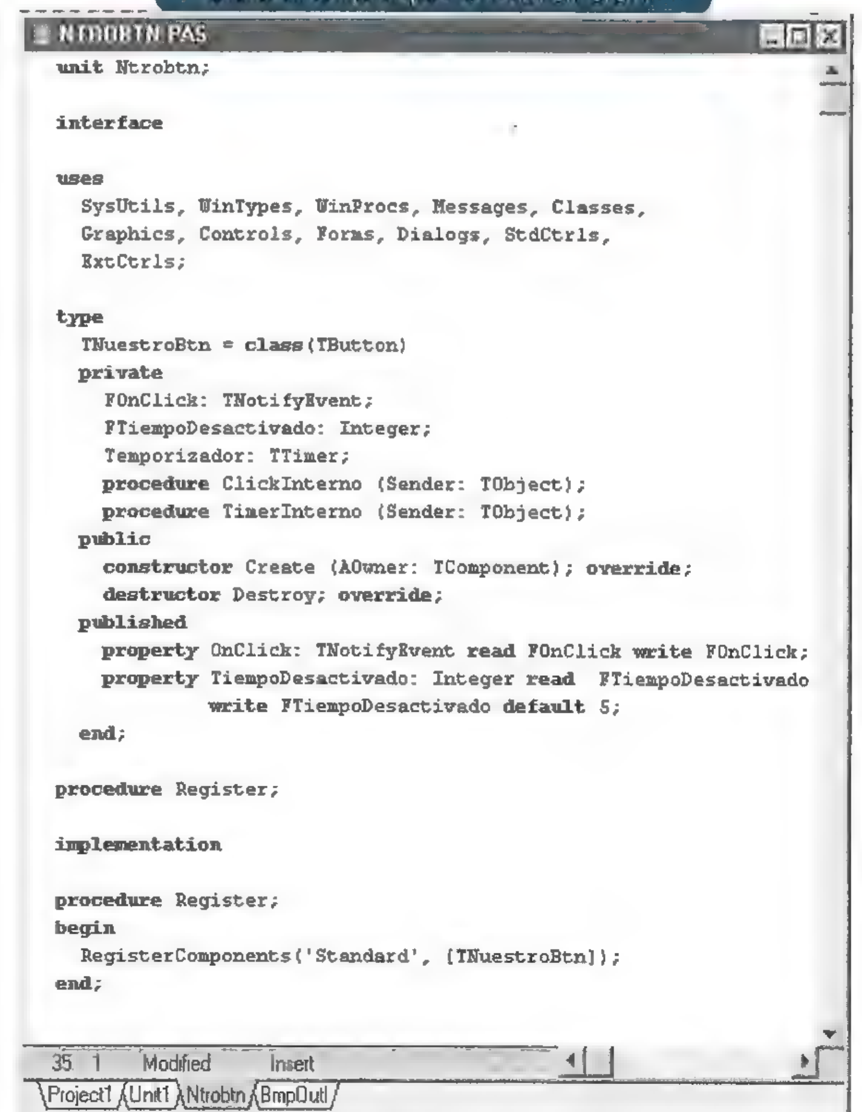
La palabra "FPrueba" de nuestro ejemplo hace referencia a una variable interna del componente que deberemos definir en la parte "private" (lógicamente para que sea interna), la cual realmente almacenará el valor de la propiedad Prueba. El hecho de que aparezca en la parte "private" prohibe al usuario del componente su acceso y modificación, así como al programador de componentes que derive su clase de la nuestra.

¿Y dónde deberemos situar el código de definición de la propiedad? Si lo que queremos es que la propiedad aparezca en el *Object Inspector* (para que pueda ser leída o modificada tanto en tiempo de diseño como en tiempo de ejecución) deberá ir en la parte *published* de la clase. Mientras que si sólo queremos que sea accesible en tiempo de ejecución, irá en la parte *public* (aunque así será memos práctica). Por último, podrá ir en la parte *protected* si nuestra intención es definir un componente que sirva de base a otros. Por ejemplo, si definimos un componente: "TAnimal", del que sólo queremos implementar la descripción general de los animales, pero del que no vamos a hacer uso directamente. Es decir, definimos "TAnimal" y de él derivamos "TGato", "TPerro", "TTrucha", "TCodorniz"... Puede existir una propiedad "Alas" común a muchos animales (las aves) pero no a todos. Al definirla en la parte protected de la clase padre "TAnimal", bastará con redeclararla como *published* en las clases hijas en las que haga falta ("TCodorniz", "TGorrión"...) y no redeclararla en el resto (por lo que para el usuario de dichas clases aparecerá como inexistente).

Una propiedad nueva

Para definir una nueva propiedad, recogeremos el componente que creamos el mes pasado. Entonces habíamos creado un componente que cambiaba la

LISTADO 1 A. Código de un componente que se desactiva durante un tiempo después de recibir un "click".



```
unit Ntrobtn;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls,
  ExtCtrls;

type
  TNuestroBtn = class(TButton)
  private
    FOnClick: TNotifyEvent;
    FTiempoDesactivado: Integer;
    Temporizador: TTimer;
    procedure ClickInterno (Sender: TObject);
    procedure TimerInterno (Sender: TObject);
  public
    constructor Create (AOwner: TComponent); override;
    destructor Destroy; override;
  published
    property OnClick: TNotifyEvent read FOnClick write FOnClick;
    property TiempoDesactivado: Integer read  FTiempoDesactivado
             write FTiempoDesactivado default 5;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Standard', [TNuestroBtn]);
end;
```

"fuente" por defecto de la clase "TButton" para que fuera *Arial*, Cursiva de 12 puntos.

Vamos a definir una nueva propiedad para nuestro botón que especifique un número de segundos durante los cuales el botón permanecerá desactivado después de hacer "click" en él. Esto puede ser útil cuando es necesario que tras hacer "click" en el botón, el usuario no pueda hacer "click" de nuevo en un tiempo durante el que el usuario podrá hacer un procesamiento determinado. El nombre que vamos a dar a esta propiedad es "TiempoDesactivado". Será una propiedad tanto de lectura como de escritura y como sólo se efectuarán acciones después de un "click" y no al modificar o leer la propiedad, no habrá funciones a las que llamar en la parte "read" y "write". Declaremos, por tanto, en la parte published:

```
Property TiempoDesactivado: Integer read FTiempoDesactivado write FTiempoDesactivado; default 5;
```

La variable "FTiempoDesactivado" será la que almacene el número de segundos que permanecerá desactivado el botón después del "click". Su declaración aparecerá en la parte private de la clase. La parte *default* proporciona el valor por defecto de la propiedad "TiempoDesactivado", fijando éste en "5". Pero para que sea efectivo, hay que añadirlo en el constructor:

```
FTiempoDesactivado := 5;
```

LISTADO 1.B. Código de un componente que se desactiva durante un tiempo después de recibir un "click".

NTROBTN.PAS

```
constructor TNuestroBtn.Create (AOwner: TComponent);
begin
  inherited Create (AOwner); {Create del padre}
  Font.Name := 'Arial';       {Fuente Arial}
  Font.Style := [fsItalic];   {cursiva}
  Font.Size := 12;            {tamaño 12 puntos}
  FTiempoDesactivado := 5;    {Tiempo por defecto: 5 segundos }
  { Inicializamos el evento OnClick de la clase padre "TButton"}
  inherited OnClick := ClickInterno;
  FOnClick := nil;
  { Creamos el temporizador y lo inicializamos }
  Temporizador := TTimer.Create (Self);
  Temporizador.Enabled := False;
  Temporizador.OnTimer := TimerInterno;
end;
procedure TNuestroBtn.ClickInterno (Sender: TObject);
begin
  { Fijamos el intervalo y activamos el Timer }
  Temporizador.Interval := FTiempoDesactivado * 1000;
  Temporizador.Enabled := True;
  Enabled := False; { Desactivamos el botón }
  { Si está asignado, devolvemos el evento }
  if Assigned (FOnClick) then FOnClick (Sender);
end;
procedure TNuestroBtn.TimerInterno (Sender: TObject);
begin
  { Activamos el botón y desactivamos el Timer}
  Enabled := True;
  Temporizador.Enabled := False;
end;
destructor TNuestroBtn.Destroy;
begin
  Temporizador.Destroy;
  inherited Destroy;
end;
end.
```

57: 1 Modified Insert

Project1 Unit1 Ntrobtn BmpOutl

Eventos

Antes de continuar es preciso hacer un inciso para introducir los "eventos". Un evento no es más que una propiedad especial cuyo valor, en lugar de ser un tipo, es un método de reacción ante un suceso. Es decir, el tipo de un evento es un "tipo función" (puede entenderse como un puntero a función). Al ser una propiedad, habrá una variable que recoja internamente el valor del evento. Esta variable será un *método*.

Por convenio, los eventos se nombran comenzando por "On" (en caso de). Ejemplos de eventos son: "OnClick", "OnMouseDown", "OnMouseUp",... La variable que recoge el valor del evento, que será un método, comenzará (como para cualquier otra propiedad) con "F". Por ejemplo, el evento "OnClick" se declarará:

```
Property OnClick: TNotifyEvent read FOnClick write FOnClick;
```

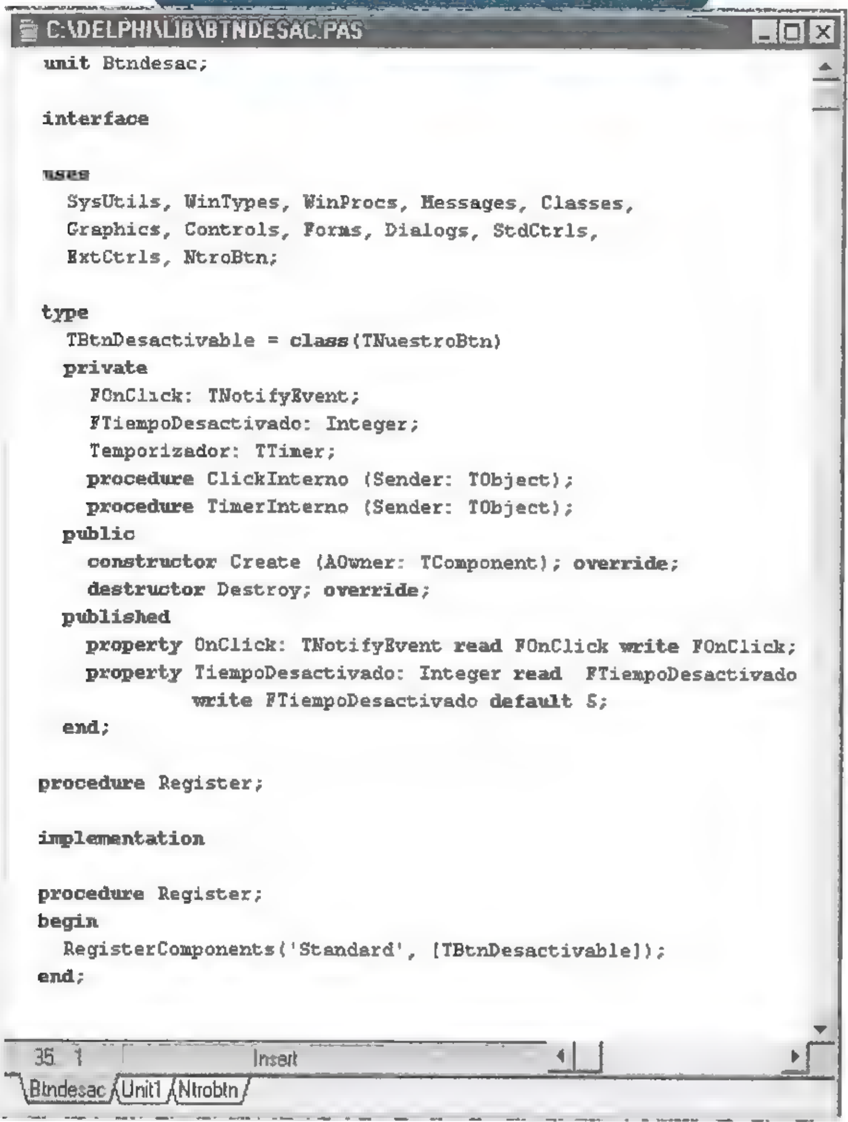
Como en nuestro componente vamos a realizar un procesamiento interno cuando el usuario genere un "click", es necesario redefinir el evento "OnClick" que heredamos de la clase TButton para poder particularizarlo a nuestros requerimientos. En la parte published escribiremos:

```
Property OnClick: TNotifyEvent read FOnClick write FOnClick;
```

y en la parte private:

```
FOnClick: TNotifyEvent;
```


LISTADO 2-A. Código del componente cuando lo heredamos de "tnuestrobtn", en lugar de heredarlo de "tbutton".




```
unit Btndesac;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls,
  ExtCtrls, NtroBtn;

type
  TBtnDesactivable = class(TNuestroBtn)
  private
    FOnClick: TNotifyEvent;
    FTiempoDesactivado: Integer;
    Temporizador: TTimer;
    procedure ClickInterno (Sender: TObject);
    procedure TimerInterno (Sender: TObject);
  public
    constructor Create (AOwner: TComponent); override;
    destructor Destroy; override;
  published
    property OnClick: TNotifyEvent read FOnClick write FOnClick;
    property TiempoDesactivado: Integer read  FTiempoDesactivado
             write FTiempoDesactivado default 5;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Standard', [TBtnDesactivable]);
end;
```

Para ello, en el constructor de la clase inicializamos el evento "OnClick" de la clase padre (TButton) para que llame a un método interno de nuestra clase (TNuestroBtn). Con esto, "desviamos" el "click" a nuestro componente para poder realizar operaciones que nos parezcan convenientes y, luego, "devolverlo" por su cauce original.

Habrá que escribir las siguientes líneas en el constructor de la clase:

```
inherited OnClick := ClickInterno;
FOnClick := nil;
```

Con la primera línea, provocamos que los "clicks" sean "desviados" a la función "ClickInterno". Si no pusiésemos "inherited", sería el OnClick que hemos redefinido quien llamaría a dicha función y, por tanto, nada funcionaría. La función "ClickInterno" irá declarada en la parte private de la siguiente manera:

```
private
procedure ClickInterno (Sender: TObject);
```

Los parámetros de esta función han de ser los mismos que los del tipo del evento que la llama. Como el evento "OnClick" es del tipo "TNotifyEvent" que se define como:

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

la función ClickInterno llevará como parámetro:

```
(Sender: TObject)
```

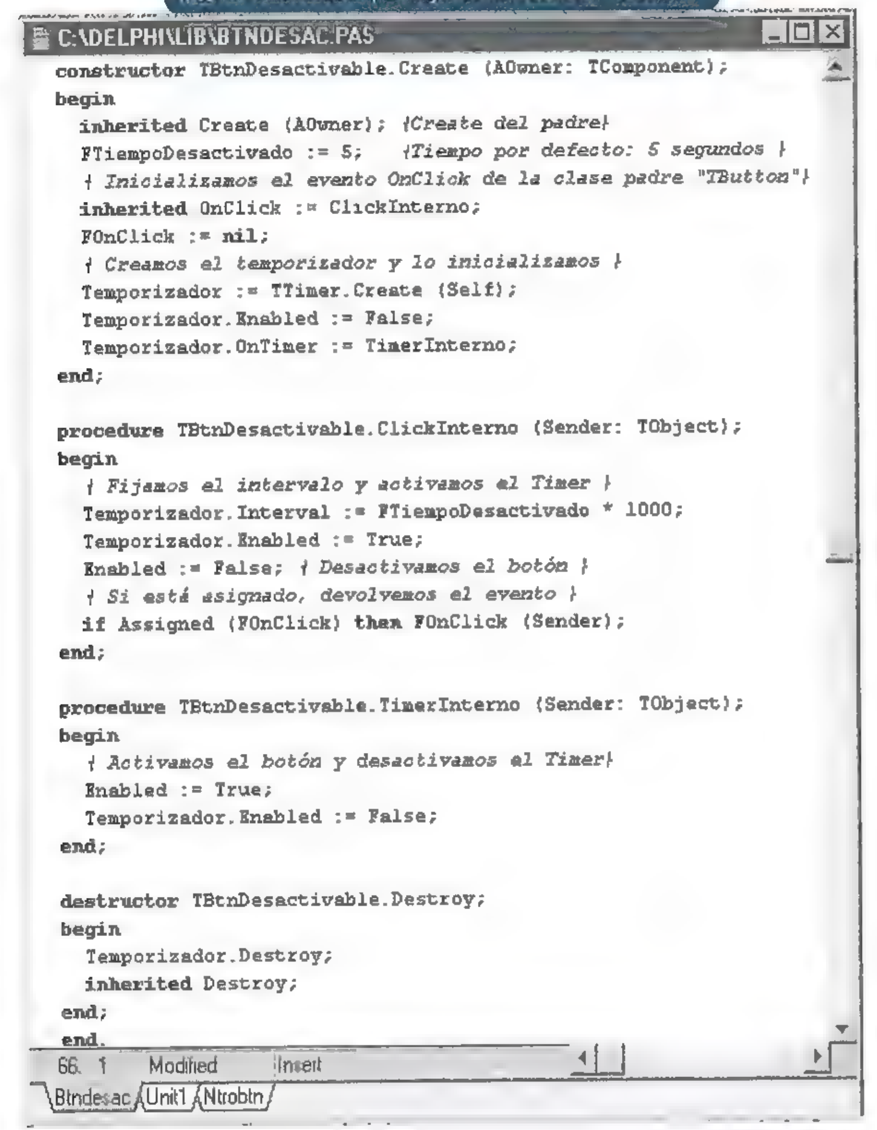
La línea: FOnClick := nil; inicializa la variable del evento redefinido a "nil" (nada), pues en un principio no habrá un método que maneje lo que ocurra tras el evento.

En este punto merece la pena destacar un aspecto. Dado que "FOnClick" es la variable que almacena el método al que hay que llamar cuando se hace "click" en un botón, el hecho de inicianizarla a "nil" significa que, por defecto, no se llama a ningún método. Y este es el funcionamiento habitual de los componentes en Delphi: por defecto no hacen nada, salvo que nosotros asignemos un método a sus eventos. Y, ¿cuál es la forma de hacerlo? Es tan sencillo que lo utilizamos a diario sin darnos cuenta: cuando seleccionamos la pestaña "Events" del *Object Inspector*, aparecen dos columnas: en la primera, se muestran los eventos que posee el componente y, en la segunda, los métodos a los que se llama cuando se produce el evento correspondiente.

Por defecto la segunda columna está vacía. Es decir, todos los eventos apuntan al "nil". Cuando hacemos "doble click" en la segunda columna, por ejemplo en la línea "OnMouseDown", automáticamente se genera un método para dicho evento y Delphi crea el código básico para manejarlo, tal y como se muestra en la figura 2.

Habíamos visto cómo manejar internamente el evento de la clase padre. Ahora descubriremos cómo "devolver" el evento para que pueda ser manejado por las clases hijas o por los objetos (instancias) de la clase.

LISTADO 2.B. Código del componente cuando lo heredamos de "tnuestrobtn", en lugar de heredarlo de "tbutton".



C:\DELPHI\LIB\BTNDESAC.PAS

```
constructor TBtnDesactivable.Create (AOwner: TComponent);
begin
  inherited Create (AOwner); {Create del padre}
  FTiempoDesactivado := 5;   {Tiempo por defecto: 5 segundos }
  { Inicializamos el evento OnClick de la clase padre "TButton"}
  inherited OnClick := ClickInterno;
  FOnClick := nil;
  { Creamos el temporizador y lo inicializamos }
  Temporizador := TTimer.Create (Self);
  Temporizador.Enabled := False;
  Temporizador.OnTimer := TimerInterno;
end;

procedure TBtnDesactivable.ClickInterno (Sender: TObject);
begin
  { Fijamos el intervalo y activamos el Timer }
  Temporizador.Interval := FTiempoDesactivado * 1000;
  Temporizador.Enabled := True;
  Enabled := False; { Desactivamos el botón }
  { Si está asignado, devolvemos el evento }
  if Assigned (FOnClick) then FOnClick (Sender);
end;

procedure TBtnDesactivable.TimerInterno (Sender: TObject);
begin
  { Activamos el botón y desactivamos el Timer}
  Enabled := True;
  Temporizador.Enabled := False;
end;

destructor TBtnDesactivable.Destroy;
begin
  Temporizador.Destroy;
  inherited Destroy;
end;
end.
```

66: 1   Modified   Insert

Btndesac   Unit1   Ntrobtn

En el método "ClickInterno", escribiremos lo siguiente:

```
procedure
TNuestroBtn.ClickInterno (Sender: TObject);
begin
 ...
 if Assigned (FOnClick)
     then FOnClick (Sender);
 ...
end;
```

Es decir: si el evento "OnClick" ha sido asignado a un método (y por tanto la variable "FOnClick" está "asignada"), entonces llamamos a ese método: FOnClick(Sender);

Fijando el temporizador

Por último, falta añadir el código que controla el tiempo que permanecerá desactivado nuestro componente después de un "click". Para lograrlo, necesitamos hacer uso de un *Timer* (Temporizador), que ya viene implementado en Delphi como un componente. Declararemos el *Timer* como una variable privada, de nombre "Temporizador":

```
private
 Temporizador: TTimer;
```

Como el *Timer* es un componente, habrá que crearlo antes de poder hacer uso del mismo. En el constructor de nuestra clase "TNuestroBtn", añadimos las siguientes líneas:

```
constructor TNuestroBtn.Create (AOwner:
TComponent);
begin
 ...
 Temporizador := TTimer.Create (Self);
 Temporizador.Enabled := False;
 Temporizador.OnTimer := TimerInterno;
 ...
end;
```

La primera línea crea el temporizador. La segunda línea prohibe, en un principio, la activación del temporizador. La tercera, asigna al evento "OnTimer" (Al vencer el temporizador) el método "TimerInterno", que habrá que definir, en la parte *private* de la clase:

```
private
 procedure TimerInterno (Sender: TObject);
```

Como al hacer "click" queremos desactivar el botón, añadiremos este código en el método "ClickInterno":

```
procedure
TNuestroBtn.ClickInterno (Sender: TObject);

begin
Temporizador.Interval :=
FTiempoDesactivado*1000;
 Temporizador.Enabled := True;
 Enabled := False;
 if Assigned (FOnClick)
     then FOnClick (Sender);
end;
```

Con la primera línea de código, convertimos las unidades de segundos a milisegundos (pues lo que maneja el *Timer* son milisegundos) y asigna ese valor a la propiedad *Interval* del *Timer* (que es la propiedad del *Timer* que informa acerca de cuánto tiempo ha de pasar, en milisegundos, para que se dispare el evento "OnTimer"). La segunda línea activa el temporizador, de forma que empiece a "contar" el tiempo

LISTADO 3.A. Código del módulo de prueba del componente



```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls, BtnDesac;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
   public
    BtnDesactivable: TBtnDesactivable;
    procedure BtnDesactivableClick(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
```


que debe transcurrir para que se dispare el evento "OnTimer"). La tercera línea es la encargada de desactivar el botón.

Una vez que pasen "FTiempoDesactivado" segundos, se disparará "OnTimer", que es manejado por la función "TimerInterno". Allí debemos introducir las líneas de código que restablecer el estado del botó:

```
procedure TNuestroBtn.TimerInterno (Sender: TObject);
begin
 Enabled := True;
 Temporizador.Enabled := False;
end;
```

Con este código activamos de nuevo el botón y desactivamos el temporizador (para que no se vuelva a disparar el evento "OnTimer" hasta que no volvamos a pulsar el botón).

Aún queda una cosa. Como el componente "Temporizador" lo hemos creado por código, también deberemos destruirlo por código. Para lograrlo, hacemos *override* del destructor de la clase base:

```
public
 destructor Destroy; override;
```

y en la parte *implementation*, escribimos el siguiente código:

```
destructor TNuestroBtn.Destroy;
begin
 Temporizador.Destroy;
 inherited Destroy;
end;
```

La primera línea es la que lleva a cabo la destrucción del temporizador, mientras que la segunda hace una llamada al destructor del padre.

El código completo del componente, se muestra en el LISTADO 1.

Dos niveles de herencia

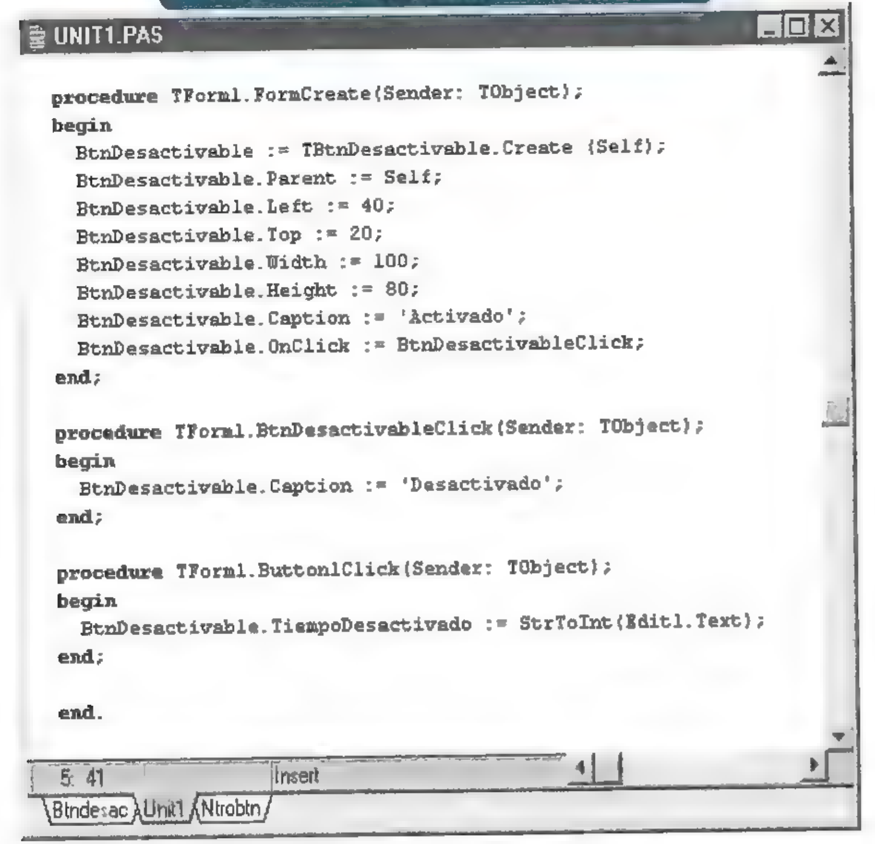
Podríamos querer que este nuevo componente fuese un tipo particular de la clase "TNuestroBtn" que definimos el mes pasado. Es decir, podríamos querer seguir utilizando la clase "TNuestroBtn" tal cual la definimos con anterioridad, y querer definir una clase nueva "TBtnDesactivable" que añadiese la funcionalidad explicada en este artículo. Para ello bastaría con declarar esta nueva clase como descendiente de "TNuestroBtn" en lugar de heredera de "TButton" y añadir en la parte *uses* de la *unit* (por ejemplo: "BtnDesac.pas") el nombre de la clase padre: NtroBtn.

El código de esta nueva clase quedaría tal cual se presenta en LISTADO 2, manteniéndose el código de "TNuestroBtn" tal cual aparecía en el artículo del mes pasado.

Módulo de prueba

Antes de instalar el componente debemos comprobar que está escrito de forma correcta. Compilaremos su código incorporándolo a un proyecto de prueba donde crearemos el componente manualmente. Para ello abrimos un nuevo proyecto (*Project1*), añadimos en la parte *uses* de la *unit1* el nombre del fichero que contiene nuestro componente (BtnDesac) y por último incorporamos en la parte *public* del formulario *TForm1* el nombre del componente:

LISTADO 3 B. Código del módulo de prueba del componente.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  BtnDesactivable := TBtnDesactivable.Create (Self);
  BtnDesactivable.Parent := Self;
  BtnDesactivable.Left := 40;
  BtnDesactivable.Top := 20;
  BtnDesactivable.Width := 100;
  BtnDesactivable.Height := 80;
  BtnDesactivable.Caption := 'Activado';
  BtnDesactivable.OnClick := BtnDesactivableClick;
end;

procedure TForm1.BtnDesactivableClick(Sender: TObject);
begin
  BtnDesactivable.Caption := 'Desactivado';
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  BtnDesactivable.TiempoDesactivado := StrToInt(Edit1.Text);
end;

end.
```

```
public
 BtnDesactivable: TBtnDesactivable;
```

En el método Create del formulario (FormCreate) escribimos lo siguiente:

```
procedure TForm1.FormCreate (Sender: TObject);
begin
  BtnDesactivable := TBtnDesactivable.Create (Self);
  BtnDesactivable.Parent := Self;
  BtnDesactivable.Left := 40;
  BtnDesactivable.Top := 20;
  BtnDesactivable.Width := 100;
  BtnDesactivable.Height := 80;
  BtnDesactivable.Caption := 'Activado';
  BtnDesactivable.OnClick := BtnDesactivableClick;
end;
```

La palabra "BtnDesactivableClick" hace referencia al método que manejará en el objeto "BtnDesactivable" (instancia de la clase "TBtnDesactivable") el evento "OnClick".

Tecleamos en la parte public:

```
public
   procedure BtnDesactivableClick (Sender: TObject);
```

y en la parte implementation:

```
procedure TForm1.BtnDesactivableClick (Sender: TObject);
begin
 BtnDesactivable.Caption := 'Desactivado';
end;
```

Por último añadimos un "TEdit" y un "TButton" para que podamos fijar en tiempo de ejecución el valor de la propiedad "TiempoDesactivado" del botón "NuestroBtn". Para ello añadimos el siguiente código en "OnClick" del botón "Button1":

```
procedure TForm1.Button1Click(Sender: TObject);

begin
   BtnDesactivable.TiempoDesactivado := StrToInt(Edit1.Text);
end;
```

El código de este módulo de prueba se presenta en el LISTADO 3.

Una vez compilado y ejecutado el proyecto, al hacer "click" en el botón se desactivará y presentará un nuevo título: 'Desactivado'.

Instalar el componente

Ya sólo queda instalarlo. Para ello, se selecciona en el menú *Options* la opción *"Install Components..."* si trabajamos con Delphi 1.0, o en el menú *Component* la opción *"Install Components..."* si trabajamos con Delphi 2.0. Pulsamos el botón *"Add..."*, introducimos el nombre del fichero que contiene nuestro componente, con la ruta completa, pulsamos *OK* y cuando termine de compilar ya tendremos el componente instalado en la paleta de Delphi.

Conclusión

En este artículo se han revisado las nociones introducidas en el artículo anterior y se han ampliado para conocer a fondo las propiedades y eventos de Delphi. Se ha visto cuáles son todos los pasos necesarios para crear e instalar un componente al que hemos añadido una propiedad. Y por último se ha creado un código para probar dicho componente antes de añadirlo a la paleta de componentes. Por último y más importante, se ha demostrado lo fácil que resulta realizar todo este proceso sin tener en ningún momento que salir del entorno de desarrollo de Delphi. En posteriores artículos se irá descubriendo cómo implementar componentes con mayor grado de complejidad y, por tanto, más adaptados a nuestras necesidades particulares. Además, se descubrirán otros secretos y trucos de este compilador, así como una comparativa entre sus distintas versiones y las mejoras realizadas en cada una de ellas.



Layer 3, MPEG rompe las barreras del sonido

Chema Álvarez y Ernesto Schmitz

Qué tendrá este formato de sonido para que grandes multinacionales de renombre como Thomson o ITT estén gastando millones de dólares en Investigación y Desarrollo de todo tipo de dispositivos basados en esta técnica de compresión

En Octubre del año pasado llegó al Cyber-Espacio un nuevo formato de sonido, podríamos pensar: vaya, otro más a añadir a la consabida lista WAV, VOC, MOD, etc..., pero quizá cuando termines de leer este párrafo, no pensarás lo mismo; si alguna vez has intentado digitalizar una canción proveniente de un CD o un disco para incluirla en una de tus DEMOS, el juego que estás desarrollando o esa aplicación multimedia que te han encargado, habrás comprobado que el espacio ocupado en el disco es prohibitivo; por ejemplo, una canción de tu grupo favorito de unos 3 minutos ocupa en formato WAV unos 30Mb lo cual es prácticamente inutilizable, pero si a ese fichero WAV le aplicamos los métodos de compresión que luego te explicaremos, con programas y utilidades SHAREWARE que incluimos en el CDROM de la revista y nos ponemos a comparar los tamaños de los ficheros obtenidos por los formatos WAV y MP3, veremos que éste último no medirá más de 3Mb.

Si las cifras no te convencen, cosa que dudamos, después de leer el artículo haz la prueba tú mismo y verás que a pesar de la compresión no hay pérdida de sonido "humanamente" apreciable. Antes de que se te quite la cara de asombro vamos a ver qué es eso de MPEG Layer-3.

Siglas

MP3 es la abreviatura de MPEG Layer-3, que es uno de los comités de trabajo del Grupo de Expertos sobre Imágenes en Movimiento (*Moving Pictures Experts Group*), de la Organización Internacional sobre Estándares (ISO) encargada de normalizar en el mundo de las empresas en general y en la informática en particular. Este organismo pone de acuerdo a los distintos fabricantes para que hagan equipos lo más compatibles entre sí (cosa que no siempre consiguen); en concreto MPEG-1 y MPEG-2 se ocupan de la manera de codificar y almacenar digitalmente películas y su sonido asociado. Dentro de estos grupos hay una parte que es la que nos interesa en el presente artículo: la dedicada al audio. En ella descubrimos las sucesivas capas, Layer-1, Layer-2 y Layer-3 con las

TABLA 1

Capa	Ratio de Compresion	Tipo de Señal	Regimen del Kbps
Layer 1	1 : 4	Estereo	384
Layer-2	1 : 6...1 : 8	Estereo	256...192
Layer-3	1 : 10...1 : 12	Estereo	128...112

TABLA 2

Calidad del Sonido	Ancho de Banda	Modo	Régimen de Kbps	Ratio de Compresión
telefonico	2.5kHz	mono	8kbps	96 : 1
mejor que onda corta	4.5kHz	mono	16Kbps	48 : 1
mejor que AM	7.5kHz	mono	32kbps	24 : 1
similar FM	11kHz	Estereo	56...64kbps	26...24 : 1
proximo a CD	15kHz	Estereo	96kbps	16 : 1
CD	>15kHz	Estereo	112...128kbps	14...12 : 1

que aumentan no sólo la calidad de sonido sino el régimen de Bits/Seg (ver Tabla-1).

Parece claro que de esta familia de esquemas de compresión la más potente es la Layer-3, de aparición más reciente y que cubre un rango más amplio de aplicaciones prácticas.

Cómo lo hace

El codificador analiza las componentes del espectro de frecuencias de la señal de sonido aplicándole un modelo psicoacústico para estimar el nivel de ruido apreciable. Todos los ficheros de las distintas capas tienen la misma información de cabecera con lo que consiguen ser compatibles hacia adelante y además soportan información asociada con el programa que los usa dentro de su cadena de bits.

Todas las capas trabajan con las mismas frecuencias de *sampleado* que son 32, 44,1 y 48kHz, siendo la de 44,1kHz la más utilizada.

El sistema de codificación consume mucha CPU pero no se hace en tiempo real, sin embargo la descodificación sí se lleva a cabo en tiempo real (reproducción de un fichero .MP3). La reproducción no consume muchos recursos, ya que, usando WinPlay3 y atendiendo a los datos proporcionados por el fabricante (aunque creemos que son incluso mejores), el programa reproduciendo una canción a 44,1kHz y 128kbps -lo cual da una calidad de CD- produce un consumo de menos de un 30% de los recursos del procesador en un Pentium-90; veamos la matriz de funcionamiento (Tabla 3).

El mejor funcionamiento de Layer-3 se debe a que la resolución de la frecuencia es 18 veces más alta que las demás, lo cual permite al codificador corregir mucho más el ruido. Además, Layer-3 utiliza codificación entrópica como el MPEG de vídeo.

Los métodos tradicionales de compresión sin pérdida de información como el LZW, Huffman, etc... normalmente no funcionan bien en la compresión de audio, como no lo hacen con la de vídeo. También existen algunos métodos de compresión con ligera pérdida de información que hay que nombrar:

Compresión de silencios (codifica los silencios numéricamente en función de su extensión).

ADPCM (Modulación por código de pulso diferencial adaptativo) tiene un rendimiento de 16..32kbps y funciona codificando la diferencia entre dos señales consecutivas adaptándola de tal manera que se usan menos bits cuando el valor es menor. Esto tiene dos inconvenientes: que hace falta prever la forma de la onda y que es propiedad de Apple; además el ratio de compresión es de 2:1.

También tenemos el LPC, código predictivo linear que ajusta la señal a un modelo de habla, que suena como la típica voz sintetizada y consigue rendimientos de 2.4kbps.

Finalmente, el CELP, que es lo mismo que el anterior pero transmitiendo un código de error y consigue rendimientos de 4.8kbps.

La otra manera de enfocar el asunto son los métodos psicoacústicos que se basan en la capacidad humana de diferenciar los sonidos, esta capacidad ronda el intervalo de 20Hz hasta 20kHz, lo que abarca un recorrido de

TABLA 3

	Pentium	486dx4-133	486dx2-66	486dx-50
MPEG-1 Estereo	si	si	*	*
MPEG-1 downmix	si	si	si	*
MPEG-1 mono	si	si	si	si
MPEG-2 Estereo	si	si	si	si
MPEG-2 downmix	si	si	si	si
MPEG-2 mono	si	si	si	si

FIGURA 1

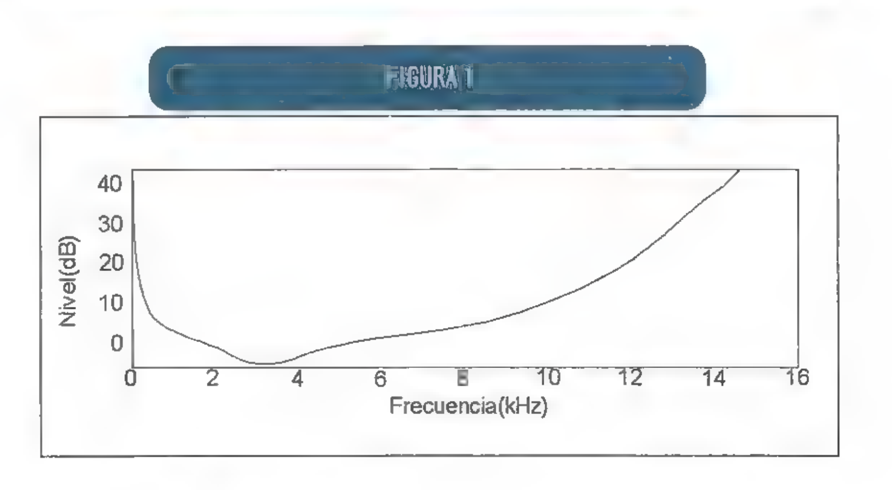


FIGURA 2

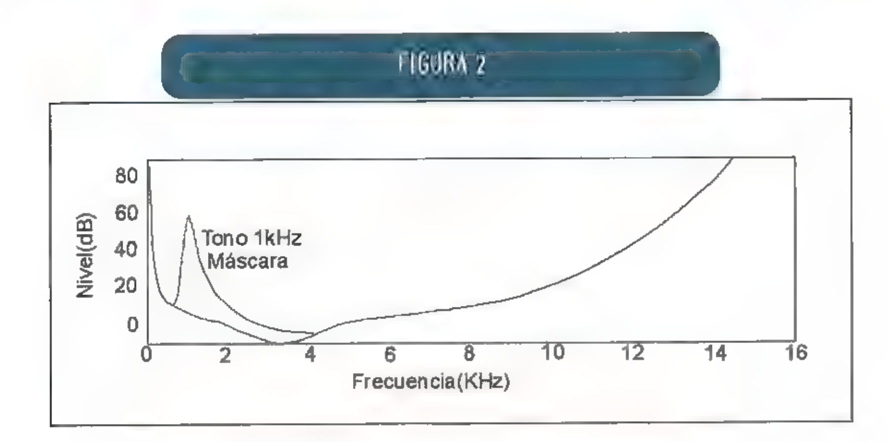
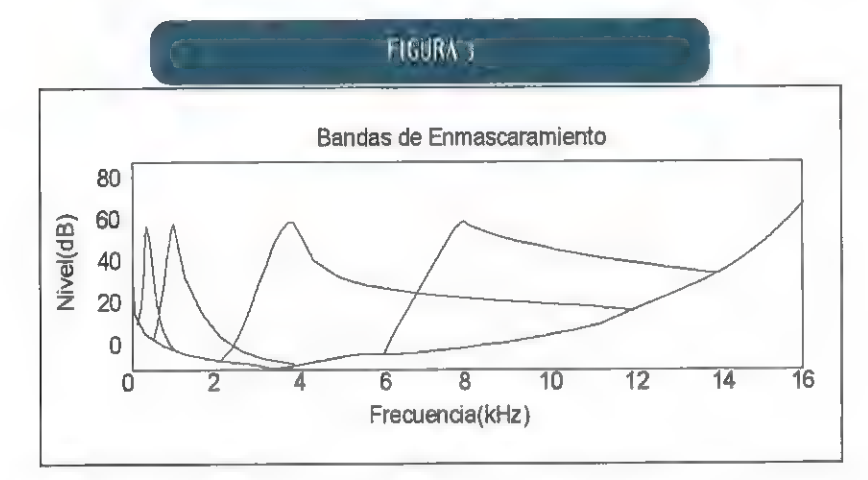


FIGURA 3



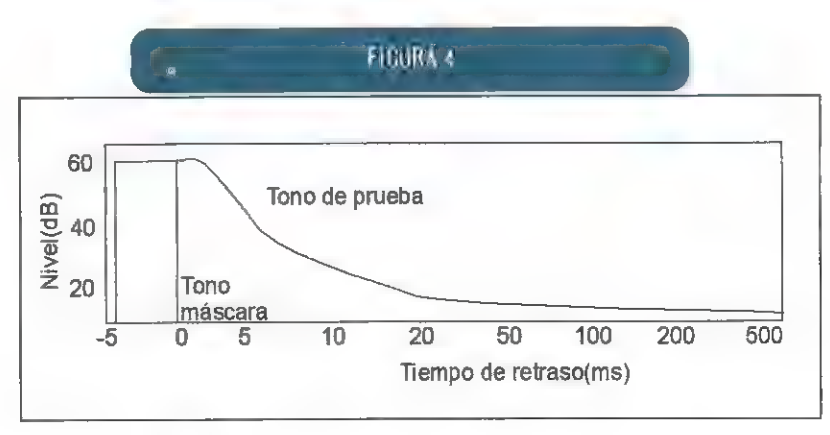
unos 96dB. Realicemos un experimento: si ponemos una persona en una habitación en silencio y la exponemos a la gama completa de frecuencias, tendremos que variar el volumen de éstas a fín de que sean percibidas por el oído humano. (Ver Figura 1).

Ahora realizaremos la misma prueba pero intentando interferir con una frecuencia máscara a un valor fijo de 60dB sobre la frecuencia de test, elevando el nivel hasta que sea distinguible dibujándola en ese momento (Figura 2).

Si repetimos la experiencia con varias frecuencias máscara, el espectro de frecuencias será como el indicado en la figura 3.

Esta enrevesada explicación lo único que nos viene a decir es que si nosotros oímos un sonido muy fuerte y se detiene de repente, tardamos un rato en percibir un sonido suave, debido a que el oído necesita tiempo en adaptarse al siguiente sonido; es decir el sonido fuerte enmascara al suave, por lo que a ojos (oídos) de un ser humano el sonido suave es superfluo. Si hacemos diferentes pruebas para ver qué retraso lleva nuestro oído a partir del tono de enmascaramiento (el fuerte) podremos realizar un diagrama con el que calcularemos el tiempo más corto a partir del cual el sonido test puede ser audible. (Figura 4).

FIGURA 4



El algoritmo

En definitiva, si oímos un tono fuerte, los tonos próximos más suaves no serán oídos, lo cual nos anima a eliminarlos. Es la llamada compresión psico-acústica en la que se basa MPEG teniendo en cuenta modelos de percepción humana del sonido para eliminar aquello no perceptible.

Veamos, MPEG-1 usa 0.3Mbps para el CD audio, que descomprimido será:

- 44,100 muestras/seg x 16bits/muestra x 2 canales > 1.4 Mbps,

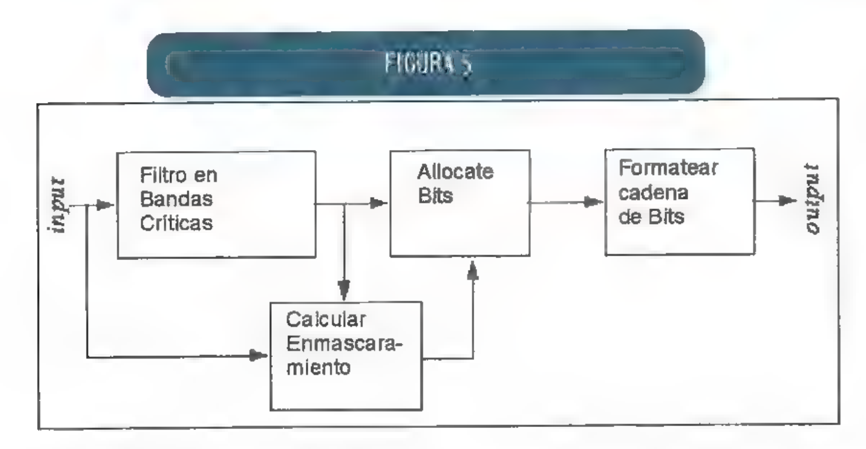
luego con un factor de compresión de 6:1 resulta en:

- 16bits x Stereo x 48kHz x 256kbps,

para lo cual se usó el grado de compresión más bajo (y de mayor calidad) disponible en layer-3. Actualmente se están haciendo pruebas en las que participan expertos escuchadores adiestrados, los cuales no fueron capaces de distinguir entre el sonido original y el comprimido.

El sistema en concreto usa unos filtros alrededor de la señal para dividirla en 32 sub-bandas de frecuencia, que usando este resultado determina el nivel de enmascaramiento que causa cada banda en la banda más cercana. Si ese nivel de enmascaramiento es mayor que una determinada cantidad, no la codifica; al mismo tiempo determina el número de bits necesarios para representar el ruido introducido por el efecto máscara

FIGURA 5



y los introduce en la cadena de bits; el diagrama de acción se muestra en la Figura 5.

Como ejemplo, vamos a analizar los primeros 16 niveles de las 32 bandas correspondientes a un instante determinado, ver tabla 4. Si nos fijamos en la banda octava veremos que el nivel de enmascaramiento apreciado en la banda séptima es de 12dB (que se calcula mediante fórmulas cuya explicación se sale de los propósitos de este artículo), lo cual es mayor que los 10dB de dicha banda, de forma que la ignoramos; en la novena pasa lo contrario, el nivel de enmascaramiento es de 15dB, menor que los 35dB de esa banda, así que no la ignoramos. Además podemos codificar el error de cuantificación con 2 Bits.

TABLA 4



Band	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Level (db)	0	8	12	10	6	2	10	60	35	20	15	2	3	5	3	1

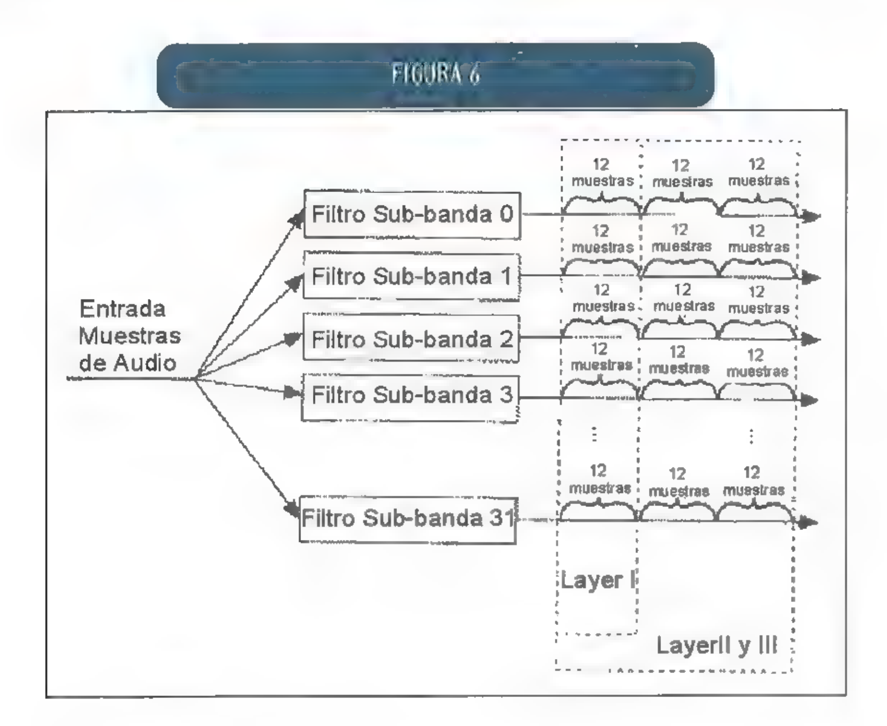
Las especificaciones del Layer-3 están en la normativa ISO-11172-3

Hemos visto el sistema a nivel básico, pero éste se complica con las diferentes capas, a modo de ejemplo mostramos un gráfico de cómo se *samplea* el sonido en cada una de las capas. (Figura 6)

El formato

Los ficheros MPEG layer-2 siguen la siguiente estructura:

FIGURA 6



una cabecera de 12 bits de sincronización y 20 de información del sistema

un CRC (Código de Redundancia Cíclica de corrección de errores) de 16 bits

después viene el llamado "bit allocation" que varía según las sub-bandas y que corresponde a 4 bits para las sub-bandas bajas, 3 bits para las intermedias y 2 bits para las sub-bandas altas

a continuación, los factores de escala codificados con 2 bits que describen la combinación que estamos usando

inmediatamente después encontramos los muestreos de sub-bandas cuantificados de acuerdo al "bit allocation" y agrupados de tres en tres, éstos se llaman "gránulos"

finalmente se añade información específica del codificador sin longitud ni contenido definida pero se supone que de utilidad para éste.

Hemos incluido una somera descripción del formato MPEG Layer-2 para que se vea la estructura genérica, el formato del MPEG Layer-3 es mucho más complicado y extenso de explicar; de todas maneras en el CDROM incluimos fuentes del codificador escritas en C para el que quiera probarlo, verá que el algoritmo de codificación es sensiblemente más complicado.

Las especificaciones técnicas del Layer-3 se encuentran en la normativa ISO-11172-3 y pueden ser solicitadas al Organismo Internacional de Estándares previo pago.

La dirección de la organización, información adicional y direcciones de interés han sido incluidas al final de este artículo.

¿Cómo crear un MP3?

Una vez abordado el aspecto teórico nos vamos a introducir en la faceta práctica, la forma de crear un fichero MP3. Para ello, vamos a describir diversas utilidades (todas ellas Shareware, las cuales han sido incluidas en el CDROM de este mes) relacionadas con este proceso.

Pongamos por ejemplo que queremos realizar en dicho formato una grabación de una canción que tenemos grabada ya sea en una cinta, en un disco de vinilo o directamente de la radio. Lo que haremos será digitalizar la fuente que previamente hemos pasado por la entrada de línea (LINE-IN) de nuestra tarjeta de sonido, en este caso, la pletina, la radio o el plato de discos. Después, mediante la grabadora de sonidos del Windows (por ejemplo), crea-

El codificador analiza las componentes del espectro de frecuencias

remos un fichero .WAV preferentemente con las siguientes características:-> 44,100 Khz, 16 bits, Stereo, PCM.

No es obligatorio utilizar estos parámetros, pero de esta manera obtendremos la mejor calidad.

FIGURA 7

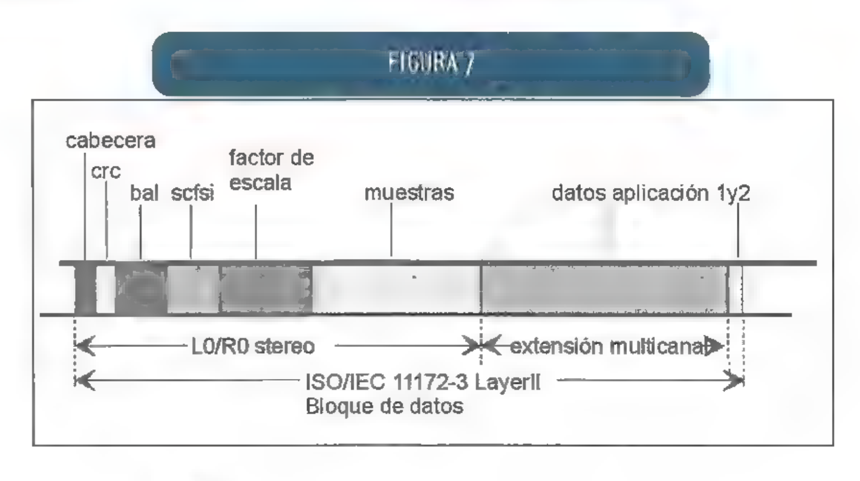


TABLA 5

Tabla 5					
MPEG-1	Layer 3	32,	44.1	48	Khz
MPEG-2	Layer 3	16,	22.05	24	Khz
MPEG-2.5	Layer 3	8,	11.025	12	Khz

Una vez tengamos el fichero, lo podremos editar a fin de eliminar aquellos trozos del principio o final que hayan sido digitalizados y no sean de utilidad, tales como silencios demasiado largos, etc. Para ello utilizaremos un editor de ficheros WAV que nos muestre la onda gráficamente; recomendamos para esta labor el CoolEdit 96 para Windows95&NT. Además de ser muy sencillo de utilizar, posee una gran cantidad de opciones avanzadas como la introducción de efectos, reverberaciones y ecos, entre otras muchas.

Habremos observado pues la gran cantidad de espacio que ocupa la digitalización, si hemos *sampleado* unos 4 minutos con las características mencionadas anteriormente (16 bits...) ésta nos estará ocupando unos 40 Mbytes. Pues bien, a este fichero resultante será al que apliquemos la compresión MPEG Audio Layer 3 mediante la utilidad L3ENC (paquete L3.ZIP incluido en el presente CD). Lo haremos mediante una orden como:

```
-> l3enc <input>.wav <output>.mp3 -crc -hq -br 112000
```

Con las opciones:

- **<-crc>** le hemos dicho que verifique errores
- **<-hq>** es para que use alta calidad
- **<-br 112000>** toma como grado de transferencia 112kbps, calidad Compact-Disc.

Recomendamos encarecidamente usar estos parámetros a menos que se disponga de poco tiempo, en cuyo caso eliminaremos la opción **<-hq>**.

La compresión de ficheros a MP3 es una tarea muy lenta, el tiempo que emplea el programa en realizar el proceso oscila como siempre en función del procesador. En un Pentium 100, un fichero de 4 minutos grabado con las anteriores características tarda en ser comprimido, tal y como hemos indicado, una media hora. Mención especial merece la opción **<-hq>**, pues añade 4 minutos más al ya de por sí largo proceso de compresión. Algunos dicen que incluso empeora las grabaciones, otros que apenas no hay diferencia, así pues lo dejamos a juicio del lector.

L3ENC soporta 3 versiones de compresión Layer-3 mostradas en la Tabla 5.

Cabe destacar la existencia de otros parámetros que pueden utilizarse disponibles sólo en la versión registrada del programa, entre ellos la posibilidad de comprimir a 128kbps, consiguiendo así la mejor relación calidad / espacio, ideal para grabaciones digitales de Compact Disc.

Sin embargo, donde de verdad se le saca partido al MPEG es en grabaciones digitales directas del CD. Anteriormente habíamos visto cómo digitalizar sonido proveniente de una fuente analógica, pues aunque lo hiciéramos desde un compact disc en vez de en la pletina, seguiríamos grabando una fuente de sonido analógica, puesto que el Compact Disc lee los datos del CDROM en formato digital y los pasa a través de un conversor digital->analógico para ser enviados al amplificador o directamente a la tarjeta de sonido. Posteriormente la grabadora de sonidos los volvía a transformar en digital.

Todo el proceso conlleva una pérdida de calidad, una dependencia del volumen de grabación, del mixer, etc... Por

Si escuchamos un tono fuerte, los tonos próximos más suaves no serán oídos

ello, si realizamos grabaciones de compact disc, conviene que lo hagamos volcando la información digital de la pista del CD a ***samplear*** directamente al disco duro sin necesidad de usar la grabadora de sonidos. Obtendremos así una copia exacta de la pista en formato .WAV a la que posteriormente aplicaremos los métodos ya conocidos para su transformación a MPEG.

Para realizar estas operaciones disponemos de varias utilidades denominadas *strippers*, la más conocida es CDDA y funciona en modo comando desde el DOS. Si lo que queremos es una herra-

TABLA 6

Tabla 6. Unidades Compatibles

Tipo-Descripción	Velocidad	Velocidad de copia	Interface	Driver
-Matsushita CR-562 (Creative CR-563)	2	2	ATBUS	Todos
-Toshiba 5301 TA	4	1	SCSI II	Adaptec
-Toshiba 3501 TA	4	1	SCSI II	Adaptec
-Toshiba 3601 TA	4,4	1	SCSI II	Adaptec
-Toshiba 5302b	4	?	ATAPI	??
-Sony CDU33A	2	2	ATBUS	??
-Sony CDU55E	2	?	ATAPI	>=2.24a
-Sony CDU55S	2	?	SCSI II	??
-Sony CDU76E	4	?	ATAPI	??
-Sony CDU76S	4	?	SCSI II	??
-Sony CDU561	?	?	SCSI	Adaptec

Definiciones:
?,?? : Valor desconocido.
ATBUS : Unidades con controladora propia.
ATAPI : Unidades que pueden ser conectadas como discos IDE.

mienta más *user friendly* optaremos por DAC, también para DOS, o CDWORX en sus versiones para Windows 95 o NT.

Un problema que nos plantea esta técnica es que no todas las unidades CDROM's del mercado soportan lectura directa, por lo que deberemos ser noso-

El sistema usa unos filtros alrededor de la señal para dividirla en 32 sub-bandas de frecuencia, que determina el nivel de enmascaramiento que causa cada banda

tros mismos quienes hagamos la prueba; hemos hallado incompatibilidades con los últimos modelos de unidades ACER y Goldstar en sus versiones IDE; al ser imposible evaluar todos los modelos, hemos incluido una pequeña tabla de las unidades compatibles más comunes que se han hallado.

De las utilidades descritas, hemos escogido CDDA al ser la más usada en Internet, puesto que las otras al ser manejadas mediante menús no precisan de una descripción detallada.

Por ejemplo, nos interesa volcar la pista número 3 de un CD al fichero TEMP.WAV, así que invocaremos el programa mediante la siguiente orden:

```
cdda /t 3 /f temp.wav /o /m
```

Con las opciones:

- **</t 3>** se informará de la pista que se va a grabar
- **</f temp.wav>** especifica el fichero destino
- **</o>** es utilizado para sobreescribir el bit de protección usado en los compact discs comerciales (Copyright)
- **</m>** hemos indicado que se use el driver MSCDEX en lugar del interface ASPI.

En el paquete donde se encuentra CDDA se incluyen otros muchos ejecutables de gran valor, en particular uno denominado CDROMINF que nos devolverá información de las características de nuestra unidad lectora CDROM así como de los drivers de ésta.

Con esta pequeña utilidad averiguaremos de una vez por todas si nuestra unidad es o no compatible con este modo de transferencia de datos.

Ahora sólo nos queda disfrutar escuchando las grabaciones que hemos realizado ya comprimidas, para ello usaremos un reproductor de ficheros MP3 como el WinPlay3 disponible para Windows 3.1 o el MuseArc en Windows 95&NT 4.0.

Este último apto únicamente para poseedores de una buena máquina.

Como colofón incluimos también aplicaciones para crear listas de reproducción (Jukebox) para estos programas, que incorporan información como nombre de la canción, autor, disco, etc...

Tenemos que advertir también que las aplicaciones Shareware WinPlay3 2.0 y Musearc 0.9 están restringidas a 20seg y 1 minuto de reproducción respectivamente. Hemos incluido además versiones Linux y Macintosh de todas estas utilidades.

TABLA 7. Direcciones de interes

http://pw2.netcom.com/~the-fly/index.html -> **The Official MPEG Audio Layer-3 Web page**

http://ds.dial.pipex.com/beast/mp3/index.htm -> **MP3 - MPEG Audio Layer-3**

http://www.iso.ch -> **Organización Internacional sobre Estándares (ISO)**

http://www.iis.fhg.de/departs/amm/layer3/ -> **Winplay3**

http://www.uni-karlsruhe.de/~ukly/ -> **Musearc**

http://www.iis.fhg.de/departs/amm/layer3/ -> **L3enc**

Utilidades MP3, JDK 1.1.1.

Demos Borland C++ Builder, Visual J++, índice, tutoriales y Shareware.

En este número hemos incluido una aplicación multimedia (autoarrancable desde Windows 95), para la instalación y visualización de los programas que hemos recopilado en el presente CDROM.

Los requisitos mínimos para la ejecución del programa son:

- Windows 95 ó NT 4.0.
- 486-6 con 8 Mb
- CDROM 6x
- una tarjeta gráfica con al menos 64K colores.

Para los que no dispongáis de este equipo, podéis acceder a los directorios de la manera habitual navegando por cada uno de los directorios del cd.

Utilidades MP3

Dentro del directorio \MP3 hemos incluido herramientas para la creación, edición y reproducción de ficheros MPEG-Layer 3.

Algunas de ellas ya han sido mencionadas en el artículo correspondiente que aparece en la revista, describiendo brevemente su manejo e instalación.

Veamos a continuación una pequeña descripción:

\DOS

- CDDA.EXE y DAC.EXE : Stripper de CDROM's

\WIN

- C96SETUP.EXE: CoolEdit 96 para Windows 95/NT 4.0
- COOL153Z.EXE: CoolEdit 96 para Windows 3.x
- COOLMP16.EXE: Filtros MPEG Layer 1 & 2 para CoolEdit (16 bits)
- COOLMP32.EXE: Filtros MPEG Layer 1 & 2 para CoolEdit (32 bits)
- MPGSRC16.ZIP: Fuentes de los filtros 16 bits.
- MPGSRC32.ZIP: Fuentes de los filtros 32 bits.
- COOLDOC.EXE: Manual CoolEdit formato Word 6.0
- CEAPI16.ZIP: Apis Cooledit para creación de filtros
- CEAPI32.ZIP: Apis Cooledit para creación de filtros
- WP200.EXE: WinPlay3 v2.0 (player de ficheros MP3) para Windows 3.x
- **\Musearc**: Musearc (otro player de ficheros MP3) para Windows 95 / NT 4.0
- CDWORX95.EXE: Stripper's de CD's para W95
- CDWORXNT.EXE: Stripper's de CD's para NT 4.0

\LINUX

- Encoders MPEG Layer 3 para Linux y plataformas NeXT

\MAC

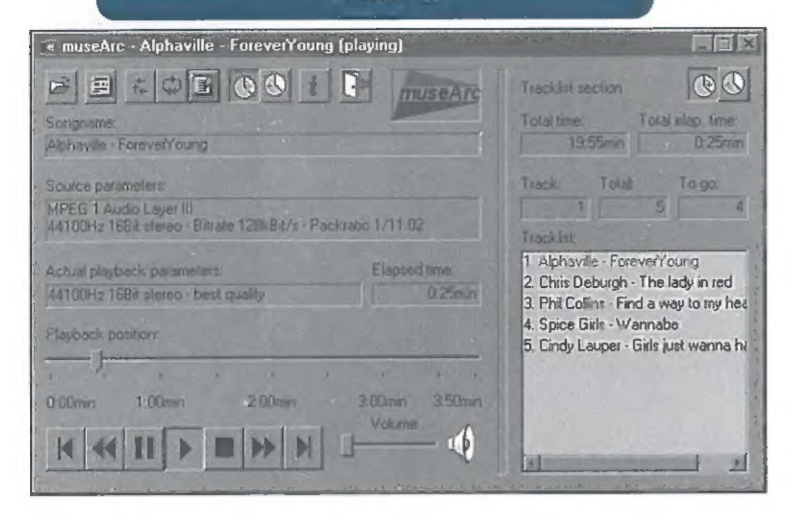
- MacPlay3 (player MP3).

Además incluimos un directorio de fuentes del formato MPEG-3 para los que quieran descubrir los entresijos de su programación.

JAVA

En el directorio \JAVA hemos incluido dos tutoriales Java: uno oficial en Inglés de Sun y otro en Castellano creado por Agustín Froufe, al cual felicitamos y damos las gracias por su colaboración.

FIGURA 1



Podéis encontrar también la última revisión del Java Development Kit de Sun (v.1.1.1) así como una trial version del Visual J++ 1.1 de Microsoft.

Es posible que en algunos sistemas no se realice la autoinstalación del JCK desde la aplicación, para conseguir su instalación deberemos ejecutar manualmente el archivo jdk1_1_1-win32-x86.exe.

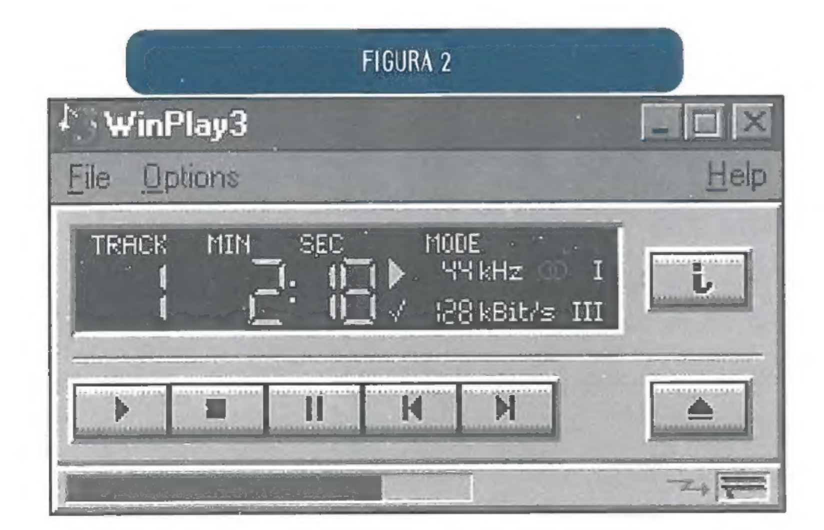
DELPHI

Primera entrega de componentes de esta herramienta de programación. Además de dos utilidades para creación de componentes para 16 y 32 bits, se encuentran toda clase de componentes para, por ejemplo, realizar la búsqueda de cadenas de texto en ficheros, búsqueda de ficheros en árboles de directorios, manejo de datos encriptados, etc... (directorio \DELPHI)

BORLAND C++ BUILDER (Trial version)

Versión de evaluación de esta poderosa herramienta de programación de Borland. (directorio \CBUILDER)

FIGURA 2



FUENTES

Como cada mes, en el directorio \FUENTES se incluyen los fuentes de la revista. Además de las correspondientes al presente número, están las del pasado mes (número 32), ya que por desgracia los usuarios de Windows 3.x se quedaron sin verlas.

ÍNDICE

Habitual de todos los CD's de la revista va a ser este índice, el cual se ha actualizado y mejorado. Como siempre, para visualizarlo será necesario poseer un navegador o *browser* de Web y abrir la página de inicio INDICE.HTM.

SOLICITUDES

Como comienzo, nos gustaría agradecer enormemente el interés de los lectores para que se incluyeran diversos programas, fuentes, shareware, tutoriales, etc. Algunas de esas peticiones ya han sido satisfechas y otras se están estudiando. Desde aquí os animamos a que continuéis enviándonos a la dirección de correo electrónico de Sólo Programadores: solop@towercom.es vuestras sugerencias para incluir en los CDs aquellos productos que os interesen. Siempre que sea posible, tendremos en cuenta vuestros deseos.

Varias de estas peticiones han hecho referencia a la inclusión de *Mirror's* (copias) de servidores Web (el de Java, Microsoft, Borland) para su visualización directa y offline a través del CD sin necesidad de soportar largas esperas o depender del tráfico en la red. Pues bien, estas peticiones están siendo estudiadas debido al descomunal tamaño de algunos servidores y a la variedad de contenidos (componentes ActiveX, VBscript, Javascript, applets Java, etc..), además de las múltiples referencias internas cruzadas necesarias.

PC Adictos

Todas las semanas tú y tu Pc tenéis una cita en La 2





Soluciones para nuestro pequeño mundo

ATENCIÓN:

INTERRUMPIMOS SU LECTURA PARA RECORDARLE QUE LAS APLICACIONES QUE HA VISTO HASTA AHORA PODRÍAN FUNCIONAR CON LAS APLICACIONES QUE VERÁ DE AQUÍ EN ADELANTE.

MQ Series es la mejor forma de hacer que aplicaciones, en diferentes entornos, se comuniquen entre sí. MQ Series está disponible en más de 20 plataformas, desde Windows NT a S/390, liberando a las aplicaciones de los distintos protocolos de comunicaciones. Para más información llame al 900 100 400 de lunes a viernes de 9 a 19h.
O consiga gratis un CD de información a través de Internet: http://www.software.ibm.com/mqseries